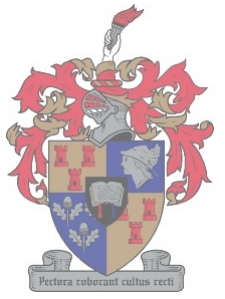
Logistieke Beplanning met behulp van Objek-Georiënteerde Simulasie

JS Loubser

Werkstuk ingelewer ter gedeeltelike voldoening aan die vereistes vir die graad van Magister in die Wysbegeerte aan die Universiteit van Stellenbosch.



Desember 1999

Studieleiers: Prof WJ Pienaar

Mnr SE Visagie

Verklaring

Ek, die ondergetekende, verklaar hiermee dat die werk in hierdie werkstuk vervat, my eie oorspronklike werk is en dat ek dit nog nie vantevore in die geheel of gedeeltelik by enige universiteit ter verkryging van 'n graad voorgelê het nie.

05/07/2019

JS Loubser Datum

Logistieke Beplanning met behulp van Objek-Georiënteerde Simulasie[1,2]

JS Loubser
Universiteit van Stellenbosch
Desember 1999

Hierdie werkstuk is opgedra aan my ouers.

---

1. Waar die *manlike vorm* in hierdie werkstuk vir enige voorwerp gebruik word, is die vroulike vorm ook geldig mits die sinsdeel dieselfde betekenis behou.
2. In die werkstuk word die verwysing na die *kliënte* van ondernemings gebruik ook vir ondernemings met *klante*. *Kliënte* is bloot die veralgemeende term omdat dit meer algemeen voorkom as *klante* in vervoerondernemings.

Opsomming

Hierdie werkstuk bied 'n uiteensetting van die gekombineerde gebruik van objek-georiënteerde programmering en simulasie, met die doel om 'n onderneming se logistieke funksies te beplan en te bedryf. 'n Gevallestudie word gebruik om die toepassing daarvan te verduidelik, en die berekeninge vir die simulasie word ondersteun deur rekenaarprogrammatuur bekend as SIMPLE++. Die program bied 'n simulasiebenadering vir die stelselnabootsing deur middel van objekformulering en interaksie tussen objekte.

'n Objekbenadering word gebruik in die analise van 'n koeriermaatskappy se ontvangste- en verspreidingsaanleg, met die oog op evaluering van die huidige stelsel. Die simulasieresultate het aangetoon dat daar 'n wanbalans in die kapasiteitsbelading van sommige werkstasies bestaan. Die maatskappy se benutting van die meeste van sy hulpbronne is redelik goed, maar deur 'n herstrukturering van aktiwiteite is dit moontlik om 'n meer doeltreffende stelsel te verkry.

Die model is die eerste elektroniese voorstelling en beplanningshulp beskikbaar van die maatskappy se aktiwiteite, en vul 'n leemte in hulle beplanningsfunksie. Die voltooide model bereik sy potensiaal wanneer dit gebruik word as strategiese operasionele gereedskapstuk, waarmee die verwagte resultate van aanpassings in die werkswyse en fisiese veranderinge voorspel kan word.

Summary

This report gives an overview of the combined use of object-oriented programming and simulation, with the objective of planning and controlling the logistic business functions. A case study to explain the application and a software package named SIMPLE++ to access the calculation is included. This package contains a simulation function to emulate the process by means of defined objects and their interaction.

An object approach is used in the analysis of the reception and distribution function of a courier company, in order to evaluate the process. Results from the simulation study showed an imbalance in the capacities of the workstations. The company's utilisation of most of its resources is fairly good, but it is possible to obtain a more efficient process by restructuring its activities.

This model is the very first electronic emulation that can be used as a planning tool for the company's business activities, and in that way it fills a gap in its current planning function. The complete simulation model only reaches its potential when used as a strategic operational tool, by forecasting the expected results of changes in the working methods and physical structure.

Bedankings

Hiermee word dank uitgespreek teenoor die personeel van 'n koeriermaatskappy in Kaapstad, wat die student van inligting voorsien het. Die gewese hoofuitvoerende beampte, mnr K Light, word bedank vir sy bereidwilligheid om die inligting en toegang tot die perseel aan die student te verleen. Mnr D Osborne en mnr J Mackay word bedank vir hulle bystand tydens die insameling van inligting.

Verdere dank word uitgespreek teenoor prof WJ Pienaar vir leiding deur die verloop van die studie, en aan mnr SE Visagie wat hom bygestaan het.

Inhoudsopgawe

Lys illustrasies

1. Logistieke beplanning

Die mobiliteit van plaaslike en internasionale markte noodsaak 'n onderneming om dienste en produkte teen kompeterende tariewe te lewer. Daar is dus toenemende druk op ondernemings om dienste en produkte van 'n hoër gehalte teen laer pryse beskikbaar te stel. Een aspek waarop ondernemings kan konsentreer vir verbeterde prosesse en laer lopende koste, is die meer doelmatige benutting van die huidige bronne tot hulle beskikking. Verskeie definisies bestaan vir *logistiek* en vir *logistieke beplanning*, maar uit dié definisies kan die volgende beskrywing saamgestel word.

1.1 Inleidende definisies

Logistiek is die proses van beplanning, implementering, en kontrolering van die doelmatige en doeltreffende vloei van roumateriaal, halfklaarprodukte, klaarprodukte, dienste, en verwante inligting vanaf die oorsprong tot by die verbruiker, met die doel om in die behoeftes van die verbruikers te beantwoord (Council of Logistics Management; 1996: 4). Hierdie definisie verwys na die inwaartse, uitwaartse, interne en eksterne beweging, en die hersirkulering van materiaal vir die beskerming van die omgewing.

Die begrip *geïntegreerde logistieke bestuur* verwys na die bestuur van die *voorsieningskanaal* of *diensteketting* ten einde die totale uitgawe aan logistieke aktiwiteite tot die minimum te beperk. Die integrasie van logistieke funksies behoort dus die geleentheid te skep om die totale logistieke koste te verminder, 'n diens van 'n hoër gehalte te lewer, en beter beheer oor die aktiwiteite van die onderneming te hê. Bestuurders kan hierdeur toesien dat minder vermorsing plaasvind, en 'n korter gemiddelde lewertyd bereik word.

Logistieke beplanning is egter meer as net beter benutting en bestuur van hulpbronne. Die denkproses waardeur logistieke beplanners moet gaan om die middele tot die beskikking van die onderneming korrek te gebruik, moet vernuwe word met die oog op die doeltreffendste benutting hiervan. Sodoende kan die minimum vermorsing

nagestreef word, en die onderneming kan die maksimum voordeel uit sy aktiwiteite verkry.

Deur deeglike analise van elke element in die logistieke diensteketting is dit vir die beplanner moontlik om die aktiwiteite meer optimaal te laat funksioneer, deur middel van kennis van die dienstempo's, opsteltye, kapasiteite, en ander belangrike karaktereienskappe van elke element. Dié beplanning moet ook gedoen word met inagneming van die kliënte se behoeftes.

1.2 Die verband tussen vervoer en logistiek

In die voorsieningskanaal dra vervoer by tot die waarde van 'n produk deur dit betyds en op die korrekte plek te besorg. Deur hierdie balansering van die vraag en aanbod van skaars hulpbronne word die eienskappe van *pleknut* en *tydnut* daaraan verleen. Voltooide produkte is van weinig waarde indien dit die mark nie betyds bereik nie, wat die belangrikheid van die vervoeraspek onderstreep. Die vervoer van voltooide produkte vanaf die oorsprong na die mark geskied deur middel van onderskeie modusse naamlik die pad, spoor, lug, pypleiding, see, en in kombinasies hiervan bekend as *intermodale* vervoer.

'n Verdere verband tussen vervoer en logistiek lê in die besluite van die onderneming wat deur die vervoeraspek beïnvloed word. So byvoorbeeld beïnvloed die ligging van ondernemings ten opsigte van ander rolspelers hulle vervoerkoste, die beskikbaarheid van sekere materiale en markte, verpakking, materiaalhantering, aspekte van kliëntediens, en tariefbepalings.

Volgens Lambert & Stock (1993) beslaan vervoerkoste ongeveer 60% van die logistieke koste binne ondernemings in die V.S.A. In Suid-Afrikaanse ondernemings beslaan die vervoerkomponent tussen 20% en 78% van die totale prys van kommoditeite. In duurder artikels is die vervoerkomponent normaalweg kleiner as dié van goedkoper artikels. Hierdie persentasies is nie noodwendig akkuraat nie, maar die groot omvang van logistieke koste toon wel aan dat doeltreffende logistieke beplanning baie belangrik is.

1.3 Belangrike aspekte van logistieke koste

Die konsep *geleentheidskoste* speel 'n groot rol in die optimale logistieke uitgawe. In die strewe na die laagste totale logistieke uitgawe as sekondêre doelwit tot die maksimering van wins, is dit soms noodsaaklik om groter uitgawes aan enkele aspekte van die stelsel aan te gaan. Dit word bereik deur op ander gebiede groter besparings te bewerkstellig. Hierdie tegniek verg goeie beplanning en moed deur die bestuur van 'n onderneming, en word soms vanuit konserwatiewe bestuurskringe as 'n risiko gesien.

'n Voorbeeld hiervan is die koste van programmatuur vir simulasiedoeleindes, wat op sigself bloot 'n kapitale uitgawe sonder direkte besparingsimplikasies is. Die resultate van bogenoemde studies werp egter in die praktyk groot vrugte af, en indien dit nie tot besparings lei nie, is die tegnologiese innovasie en elektroniese voorstelling van die werklikheid vir die meeste ondernemings van groot belang. Die koste van tegnologiese vooruitgang in hierdie veld blyk baie hoog te wees, maar in 'n kompeterende samelewing behoort ondernemings versigtig van die beskikbare tegnologie gebruik te maak, mits dit geregverdigde kort- of langtermynvoordele bied. Ondernemings moet waak teen vir 'n beheptheid met tegnologie, en verseker dat alle gevorderde apparaat wel 'n lewensvatbare doel dien.

Ander voorbeelde van geleentheidskoste-implikasies is die vervoer van goedere teenoor die aantal pakhuise, eie vervoer teenoor professionele vervoer, vervoer op sigself teenoor kliëntediens, vervoer teenoor voorraadopberging en vervoerwyse teenoor verpakking. Die keuses behoort volgens die goue reël gemaak te wees naamlik dat ondernemings op hul *kernbesigheid* moet fokus, en ander aktiwiteite waar moontlik moet uitkontrakteer. Die uitkontraktering moet gedoen word ten einde 'n meer doelmatige stelsel te bereik.

Ondernemings moet ook waak teen die uitbreiding van hulle dienste bloot ter wille van groei, terwyl hierdie uitbreidings ondoeltreffende meulstene word wat die onderneming finansiële skade kan berokken. Soms wil ondernemings addisionele dienste aan kliënte bied om die kwaliteit en waarde van hulle dienste te verhoog, maar

hulle beskik nie oor die nodige infrastruktuur en kennis om dit ekonomies lewensvatbaar te bedryf nie.

Verdere riglyne dui op die benutting van die kapitale bates tot die volle potensiaal van elke item soos voertuie, geboue, hanteringstoerusting, en rekenaars. Elk van bogenoemde items behoort as noodsaaklike bate aangekoop en goed in stand gehou te word, en die bestuur behoort te waak teen items wat bloot die beeld van die onderneming bevorder.

Ondernemings besit volgens Cronje (1996) 'n eie voertuigvloot om hoofsaaklik twee redes. Eerstens verkies die onderneming om beheer te hê oor die diensvlak en algehele vervoerfunksie, en tweedens om voordeel te trek uit die oënskynlik laer vervoerkoste. Laasgenoemde oorweging behoort deeglik ondersoek te word aan die hand van die geleentheidskoste, aangesien ondernemings dikwels nie al die relevante koste in ag neem in die berekening nie. Die gevaar bestaan dat slegs die *waargenome koste* van sommige vervoerfunksies vir hierdie berekening gebruik word, wat kan lei tot verliese in hierdie funksies. Net so kan die onderbenutting van voertuie lei tot 'n laer inkomste uit vervoer as wat aanvanklik beplan is. Lomp voertuigvlote behoort omskep te word in 'n doeltreffende onderneming waarin alle voertuie 'n hoë benuttingsvlak en kort ledige tye het.

Die vorming van vennootskappe in die totale logistieke voorsieningskanaal kan ondernemings bevoordeel, veral ten opsigte van die doeltreffendheid van gesamentlike aktiwiteite, en gehalte van die gelewerde diens. Suksesvolle vennootskappe vereis goeie kommunikasie, wedersydse eerlikheid, toewyding, 'n hoë mate van buigsaamheid, en versoenbare inligtingstelsels.

In die literatuur is daar al heelwat melding gemaak van en uitgebrei oor die omvang van inligtingstelsels, maar hierdie aspek kan nie oorbeklemtoon word nie. Die vloei van data binne die onderneming en tussen ondernemings moet glad verloop, vertroulik wees, en ook betroubare resultate bied. Programmatuur behoort 'n aanwins vir die onderneming te wees deurdat dit aktiwiteite na behore ondersteun en kontroleer. Stelsels moet aanpasbaar ontwerp word, sodat aanpassings aan die onderneming se bestaande aktiwiteite deur dieselfde stelsel hanteer kan word.

In die meeste moderne stelsels gaan voorraadvloei gepaard met inligtingvloei. Waar daar in die verlede baie op voorraadvloei en kapasiteitsbalansering gekonsentreer is, behoort inligtingvloei nou ook optimaal beplan en bedryf te word om bottelnekke te voorkom. Inligting wat tipies is van die vervoeronderneming sluit in die vragvolumes, die verskeper se prestasie, vervoerkoste per tydeenheid, aantal eenhede vervoer per tydeenheid, gereeldheid van 'n diens, bedryfskoste, en hanteringsdigtheid (Cronjé, 1996).

Ondernemings het 'n keuse uit 'n magdom tegniese funksies, wat daagliks deur nuwe tegnologie verder verbeter en verfyn word. Selektiewe benutting van die beskikbare tegnologie is dus noodsaaklik, en ondernemings moet verseker dat elke funksie bruikbaar en voordelig is met inagneming van die plaaslike toestande. Die bestuur behoort 'n visie te hê en 'n ondersteunende werksplan te vorm, en die onderneming behoort hiervolgens bedryf te word.

2. Objek-georiënteerde programmering

2.1 Agtergrond

'n Aantal terme en begrippe word in hierdie afdeling verduidelik. Die bespreking van die gekose vervoermodel geskied in hierdie werkstuk vanuit die objek-georiënteerde perspektief, waarin hierdie terme en begrippe deurgaans gebruik word.

2.2.1 Objek-oriëntasie

Objek-oriëntasie word dikwels beskou as 'n rewolusionêre nuwe tegniek. Dit is egter 'n gevestigde tegniek wat sedert die sestigerjare in gebruik is. Die eerste tekens hiervan kan gevind word in Noorweë, waar die voorloperprogram Simula reeds in 1967 hiervan gebruik gemaak het (Jacobson; 1995: 45). Die ontwikkeling van objek-georiënteerde (OG) stelsels het vanuit die rekenaarwetenskap plaasgevind. Hierdie benadering is in sy kinderskoene sedert die laat jare sestig deur programmatuur soos Simula en Smalltalk en vandag deur tale soos TRELLIS, GALILEO en C++ ondersteun (Dittrich; 1991: 100). Die beginsels en konsepte is lank voor die koms van die persoonlike rekenaar gevorm, en bestaan vandag steeds as riglyne.

Meer onlangs het die benadering in omvangryke toepassings van inligtingstegnologie en in die bestuur van databasisse meer potensiaal verwesenlik. Hedendaagse vereistes van rekenaargesteunde ontwerp en inligtingstelsels, tel onder die groot aantal redes vir die belangstelling in hierdie veld (Kim & Lochovsky; 1989: ix).

Volgens Salvendy (1992: 602) is OG programmering die modelwoord *vir kunsmatige intelligensie*, wat 'n uitstekende datastruktuur vir die simboliese manipulasie van konseptuele inligting voorstel. Die meeste konsepte van OG programmering word tans deur die rekenaartaal SMALLTALK ondersteun. Met hierdie tegniek kan die eienskappe van afgeleide objekte geïgnoreer word, aangesien die oorspronklike objekte hulle eienskappe aan die nuutgevormde objekte oordra. Die OG proses

gebruik die tegniek van *raamvoming* waarvolgens homogene rame aan mekaar geskakel word deur die programmeerder soos hy dit verkies, maar is aansienlik meer buigsaam vanweë die eienskappe van *modulariteit*, *oorerflikheid*, en *oordrag*.

Die proses van OG programmering het ontstaan in die ingenieurswese, en word vandag veral in die vervaardigingsektor gebruik. Daar bestaan vele ander toepassingsmoontlikhede hiervan, waaronder die logistieke beplanningsfunksie in 'n voorsieningskanaal. Dit is byvoorbeeld 'n ideale tegniek vir die analise van die onderskeie homogene vervoeraktiwiteite binne 'n onderneming, aangesien die doelmatigheid van elke aktiwiteit hiermee indiwidueel ontleed kan word. Dit is ook moontlik dat die aanpassings op grond van indiwiduele aktiwiteite beoordeel kan word met inagneming van die totale *uitsetsnelheid* en die doelmatigheid van die hele stelsel.

2.1.2 Objekte

'n *Objek* is enige gebeure wat 'n betekenisvolle bydrae lewer in die ondernemingsmodel, en wat opgeneem word in die modelomgewing. Elke objek beskryf 'n indiwiduele funksie wat in 'n model uitgevoer word, en wat normaalweg verband hou met ander funksies. Inligting rakende objekte word vasgevang in *attribute* tot die betrokke objek. Die versameling objekte in 'n stelsel vorm die omgewing. Die term *attribute* word in die plek van *eienskappe* gebruik, aangesien die program hiervoor voorsiening maak in 'n doelgemaakte programvenster.

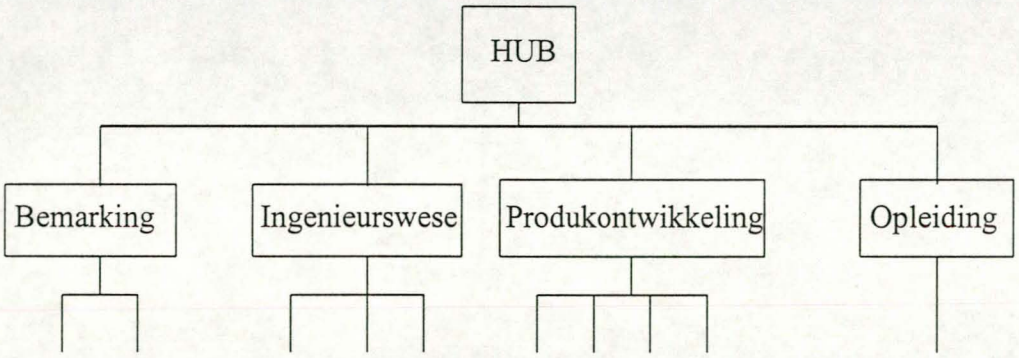
Ontwerpers van voertuie, argitekte, en siviele ingenieurs bou skaalmodelle ten einde hul ontwerpe te evalueer. Net so gebruik fisici, chemici, en wiskundiges vergelykings en formules om hulle prosesse verstaanbaar te maak. Modellering word gebruik as die saamgestelde term vir die beskrywing van wat die navorser doen. Die doelwit van 'n model is om aan 'n persoon of groep mense een of meer aspekte of perspektiewe van 'n proses duideliker te maak (Jacobson; 1995: 28).

Aandeelhouers en personeel kry verskillende modelle van 'n onderneming, wat ook verskil van die model wat kliënte kry. 'n Ondernemingsmodel toon die omgewing en

die interaksie tussen die omgewing en die onderneming aan. Die omgewing verwys in hierdie verband na enige ander instansie waarmee die onderneming kommunikeer of sake doen. So 'n model is nie noodwendig enkelvoudig nie, maar kan 'n aantal geïntegreerde modelle verteenwoordig.

'n Voorbeeld van 'n ondernemingsmodel se hiërargiese struktuur word in Figuur 2.1 gegee. Dit is een van die algemeenste vorms van die ondernemingsmodel, en is verskillend van die modelle waar prosesse van 'n onderneming herontwerp word. In laasgenoemde geval toon diagramme die aktiwiteite aan en nie die funksies soos hieronder nie. Figuur 2.1 is onvolledig, en toon slegs die boonste helfte van 'n struktuur aan.

Figuur 2.1 'n Voorbeeld van 'n ondernemingstruktuur



Normaalweg is daar 'n traagheid teen strukturele en prosedurele veranderinge in 'n onderneming, as gevolg van mense wat in hul rolle ingroei en ingelig wil bly oor die gevolge van veranderinge. Dus veroorsaak dit soms dat ondernemings net verander en aanpas wanneer dit nodig is, en die kompeterende voordeel van markgerigte aanpassings sodoende verlore gaan. Deur doeltreffende ondernemingsmodellering is dit moontlik dat vinnige aanpassings en voldoende vooruitbeplanning gedoen kan word om die gevolge van veranderinge te kontroleer, en beheer te behou oor die onderneming. Die onsekere uitkomste word sodoende beperk en deur realistiese en betroubare vooruitskattings vervang, sodat ondernemings veiliger strategiese beplanning kan doen.

Rolspelers wat 'n ondernemingsmodel kan besit, sluit kliënte, vennote, uitvoerende bestuurslede, die herontwerpspan, die proseseienaar, die hulpbroneienaar, die afdeling informasietegnologie, en menslike hulpbronontwikkeling in (Jacobson; 1995: 40). Indien hierdie partye nie elk behoorlik ingelig word oor strategiese kwessies rakende die ondernemingsmodel nie, kan die ondernemings skade opdoen as gevolg van sekere partye wat nie die veranderinge kan hanteer nie.

OG benaderings word op heelparty wyses gedefinieer, en kan ook toegepas word op 'n breë spektrum van die wetenskap. Een eienskap van die benadering kom duidelik na vore in die meeste toepassings, naamlik die vorming van interafhanklike eenhede, oftewel die verdeling van 'n komplekse stelsel in sy eenvoudigste elemente. Sodoende kan die boublokkies elk ontleed word om sy indiwiduele bydrae en prestasie te beoordeel. Die ontleders kry ook so die kans om die verwantskappe tussen die elemente te ondersoek, en beter begrip van die stelsel se werking te kry (Kim & Lochovsky; 1989: ix).

OG programmatuur poog om die verbrokkeling van komplekse stelsels formeel te kontroleer, en om die geleentheid tot verdere analise van die elemente te bevorder. Aangesien die proses van dienslewering in sy eenvoudigste elemente opgebreek word, kan elke aktiwiteit indiwidueel beoordeel word met behulp van 'n voordeelkoste-ontleding. Net so kan die elemente se karaktereienskappe gemanipuleer word om 'n meer aanvaarbare uitset te gee.

Objek-oriëntasie is 'n spesiale benadering wat meer komplekse sisteme kan modelleer. Die aktiwiteite en hulle interaksievorme word as objekte beskou, en hulle eienskappe is die attribute van die objekte. In die geval waar die aktiwiteit van een objek 'n ander objek beïnvloed, vind *kommunikasie* tussen objekte plaas (Jacobson; 1995: 45).

OG programmering is egter eers sedert 1990 op groter skaal toegepas. Tans word slegs 'n fraksie van prosesontwikkeling met hierdie tegniek gedoen, maar daar is 'n definitiewe neiging na die verhoogde gebruik van OG gereedskap binne groot ondernemings. Sedert die sewentigerjare het ontwerpers van die stelsels erken dat die moontlikheid bestaan om *apparatuur* (hardeware) met behulp van *programmatuur*

(sagteware) te modelleer, selfs sonder gevorderde programmatuur. Jacobson het in 1992 getoon dat die tipe programmeringstaal nie in modellering-programmatuur die belangrike aspek is nie, maar wel die aanpasbaarheid en verstaanbaarheid van die model. Dit het hy aangetoon deur 'n onafhanklike tegniek te ontwikkel wat modellering-programmatuur ontwerp, wat later in 'n saamgestelde taal opgeneem is (Jacobson; 1995: 46).

Net soos hierdie studie later toegespits word op die simulasietegniek, is die vroegste bekende gebruik van objek-geöriënteerde tegnieke juis uit 'n simulasiestudie in 1973 ontdek. Die studie is onderneem met behulp van rekenaarprogrammatuur genaamd Simula, 'n rekenaartaal wat spesiaal vir hierdie doel ontwikkel is. 'n Interessante verskynsel is dat die gebruik van gesimuleerde objekte 'n goeie basis bied vir die ontwikkeling van *prototipes*. Die ontwikkeling van die program Smalltalk deur Goldberg en Robson, was juis hierop gerig (1983).

2.2 Homogene stelsels

Die definisie vir OG programmering word ook op 'n eenvoudiger wyse beskou as enige meganisme in programmatuur wat voorsiening maak vir die vorming van 'n stelsel, deur dit vanuit eenvoudige boublokkies saam te stel, en as sodanig te beskou. Hierdie definisie laat dus ruimte om die mate waarin objek-georiënteerde programmering gebruik word te erken, en nie net óf dit gebruik word nie. Die mate van homogeniteit van 'n model bepaal dus bogenoemde teenwoordigheid.

'n *Homogene* model bestaan reeds in sy eenvoudigste vorm, en dien as boublokkie in die analise, terwyl 'n aantal homogene elemente saam 'n groep objekte vorm wat ontleed kan word met behulp van verskeie tegnieke. Die navorser moet deur 'n proses van *konseptualisering* gaan om die modelelemente te identifiseer, en die model te genereer.

Die tegnieke van hierdie programmeringsvorm skep 'n visuele intervlak vir 'n groep aksies in die vorm van 'n objek, waarbinne die inligting en operasionele karakteristieke versteek word. Volgens Kim en Lochovsky (1989: 5) word die

onafhanklikheid van objekte beklemtoon deur die kommunikasie tussen objekte te beskryf as die oordrag van boodskappe. Net soos die programmatuur van die tagtigerjare, gebruik weergawe 5.0 van SIMPLE++ ook die beginsel dat hierdie kommunikasie 'n paradigma, is en nie 'n strategie nie. Die programmeerder manipuleer nie die versteekte inligting nie, maar stuur wel 'n boodskap na die objek, wat dan op sy beurt 'n geprogrammeerde *metode* raadpleeg vir 'n toepaslike reaksie op die boodskap (Kim & Lochovsky; 1989: 4).

Die interaksie tussen 'n aantal metodes en die objekte word bepaal deur al die moontlike aksies wat objekte kan uitvoer, te koppel aan die ooreenstemmende metode. Sodoende kan die programmeerder die proses manipuleer, en verskillende moontlike reaksies op boodskappe in dié metodes programmeer, wat dan op hulle beurt deur die reaksie van 'n boodskap opgeroep word. Verskeie objekte kan ook dieselfde metode raadpleeg vir hulle reaksies, en sal dus dieselfde reaksie op een boodskap toon.

Met so 'n omvattende voorstelling van 'n proses is dit moontlik om voordeel te trek uit die oordrag van karakteristieke vanaf die oorspronklike objek na die *afstammelinge*. Die insluiting van sleuteleienskappe vir 'n *ras* objekte, maak dit moontlik om die afstammelinge daarin te manipuleer deur die eienskappe van die oorspronklike objek te verander om by elke situasie te pas. Die programmeerder beskik dus oor die vermoë om objekte binne en buite 'n ras te ontwikkel, om net sekere objekte as afstammelinge te behou, of andere indiwidueel of binne unieke rasse te ontwikkel. *Gedragsdeling* en verkorte reaksietye spruit hieruit, wat programmeringstyd en skakeling met soortgelyke modelle vergemaklik.

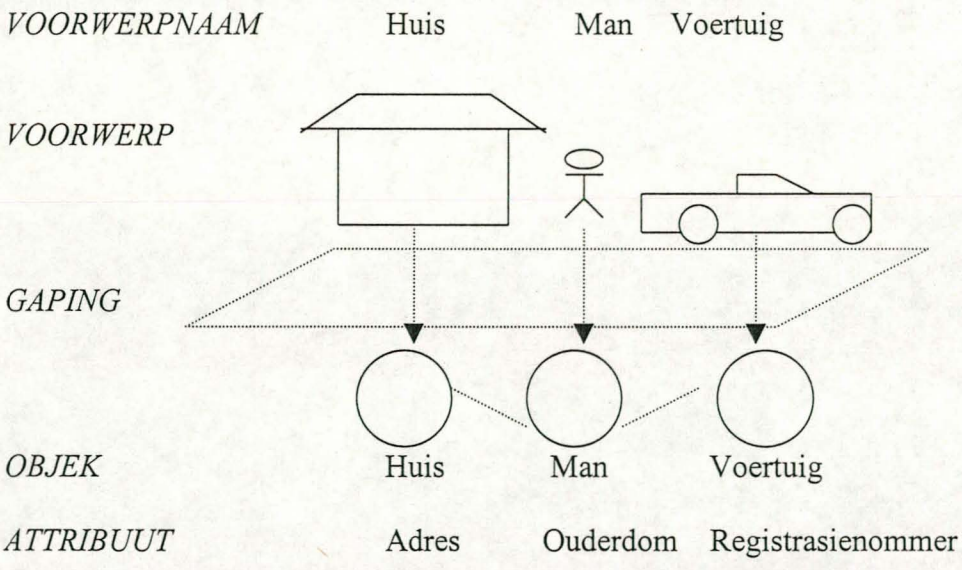
2.3 Objek-georiënteerde modelle

Die toepassing van databasistegnologie op rekenaargesteunde ontwerp, inligtingstelsels, programmatuur-ingenieurswese, en kantooroutomatisasie is volgens Dittrich (1991: 13) 'n aktiewe area in databasisnavorsing. Baie van hierdie modelle word opgebou uit komplekse en gestruktureerde objekte, wat nie maklik sonder 'n rekenaar hanteerbaar is nie.

Die aard, tydsbestek, en metode waarvolgens maatskappye of ondernemings funksies uitvoer, word in 'n ondernemingsmodel beskryf. Die model lê klem op die argitektoniese eienskappe van die onderneming, maar beskryf ook die onderlinge aktiwiteite en werkswyses. Ondernemings hanteer werklike situasies en gebeure, produseer uitsette, en verwerk insette. Hierdie in- en uitsette het 'n waarde gemeet aan die koste of pryse daarvan, en 'n kliëntediens tevredenheidsfaktor wat normaalweg meetbaar is.

Al die bogenoemde aktiwiteite kan as objekte in 'n model voorgestel word. Aangesien daar 'n noue verwantskap tussen die objekte en die werklike aktiwiteite bestaan, is die *semantiese* gaping tussen die model en die werklikheid baie klein. Figuur 2.2 toon hierdie begrip skematies aan.

Figuur 2.2 'n Semantiese gaping tussen voorwerpe en hul simboliese voorstelling



Dit is nie net moontlik om gebeure as objekte uit te beeld nie, maar verwantskappe tussen gebeure kan ook as assosiasies tussen objekte aangetoon word. 'n Normale redenasie ondersteun die natuurlike verskynsel van die voorstelling van statiese strukture van spesifieke gebeure, as objekte. Objek-oriëntasie neem hierdie redenasie een stap verder deur gebeure wat dinamies verander ook te modelleer. Sodoende stel objekte gedrag voor wat beïnvloed kan word deur die omgewing waarin dit geplaas

word. Objekte kan dus geaktiveer word, verander, en ander objekte aktiveer (Jacobson; 1995: 46).

Die sterk ooreenkoms tussen objek-georiënteerde modelle en die werklikheid, gereflekteer in die klein semantiese gaping, maak hierdie modelle maklik verstaanbaar en interpreteerbaar. Dit is een van die grootste voordele van OG modellering.

2.4 Herhaalbare gebruik

Die insluiting van eienskappe vir groepe objekte in objek-georiënteerde programmering, is jare lank ingespan om die hergebruik van gesimuleerde prosesse moontlik te maak. Die kuns om objekte vir maklike en toepasbare hergebruik te stoor, het hieruit ontwikkel. Die programmeerder poog om 'n basis van aktiwiteite en objekte in die geheue van die programmatuur te stoor, waarmee die aanvanklike modellering van verskeie prosesse kan geskied. OG programmering ondersteun vier unieke objekfunksies, naamlik:

1. Onttrekking of *abstraksie* waarvolgens 'n klas ontwikkel word wat die eienskappe van die objek definieer;
2. *Omhulselvorming* om die eienskappe van 'n objek;
3. *Oorerflikheid* oorgedra van die klas na die subklas, wat een vlak laer in die hiërargiese stelsel is; en
4. *Polimorfismes* laat die ontwerper toe om dieselfde boodskap aan verskillende objekte in hul onderskeie hiërargiese klasse oor te dra.

'n Ander programmatuurtegniek vir die ontwerp van vervaardigingstelsels is bekend as kennis-gebaseerde of *deskundige stelsels* (Salvendy; 1992: 352). Vir die ontwikkeling van so 'n stelsel moet die ontwerper die volgende belangrike aspekte in gedagte hou:

- identifisering,
- konseptualisering,
- formulering,
- evaluering,
- implementering, en

- toetsing.

Die ontwerper sou tipies 'n komplekse proses modelleer deur eerstens die belangrikste aspekte daarvan uit te lig, nuwe idees daarvoor te formuleer, en die model daarop te bou. Dit is belangrik dat die eindproduk deeglik getoets word, aangesien indiwiduele aktiwiteite soms beter resultate lewer terwyl die gesamentlike funksie verswak word.

2.4.1 Oombliklikheid en objekklasse

Een van die bekendste meganismes wat die hergebruik van basiese modelle moontlik maak, is *oombliklikheid.* Objek-georiënteerde tegnieke voorsien normaalweg ingeboude datatipes, wat staties of dinamies in die model geaktiveer kan word (Kim & Lochovsky; 1989: 5). Statiese datatipes word in die aanvang van simulasie geaktiveer en behou die oorspronklike parameters vir die tydsduur van die simulasie; dinamiese datatipes word terwyl die simulasie plaasvind op spesifieke tydstippe geaktiveer of deaktiveer.

Nou word programmeerders voor die uitdaging gestel om hulle eie objekte te ontwikkel. Hieruit het objekklasse ontstaan, wat elk 'n groep sigbare aktiwiteite, versteekte veranderlikes uniek aan die objek en versteekte metodes insluit. So byvoorbeeld is 'n groep kliënte van 'n vervoerkontrakteur almal in dieselfde klas, naamlik kliënte.

Elke objek behoort tot 'n klas, wat spesifiseer waarvoor die objek gebruik kan word. Die klas bepaal dus die aktiwiteite wat op die objek uitgevoer kan word, en die attribute van hierdie objek. Elke objek beskik oor inligting rakende sy klas en neem verskillende waardes vir die attribute aan in sy leeftyd. Hierdie veranderinge vind plaas as aktiwiteite sy attribute manipuleer. Dieselfde aktiwiteite is algemeen vir alle objekte wat tot dieselfde klas behoort (Jacobson; 1995: 63).

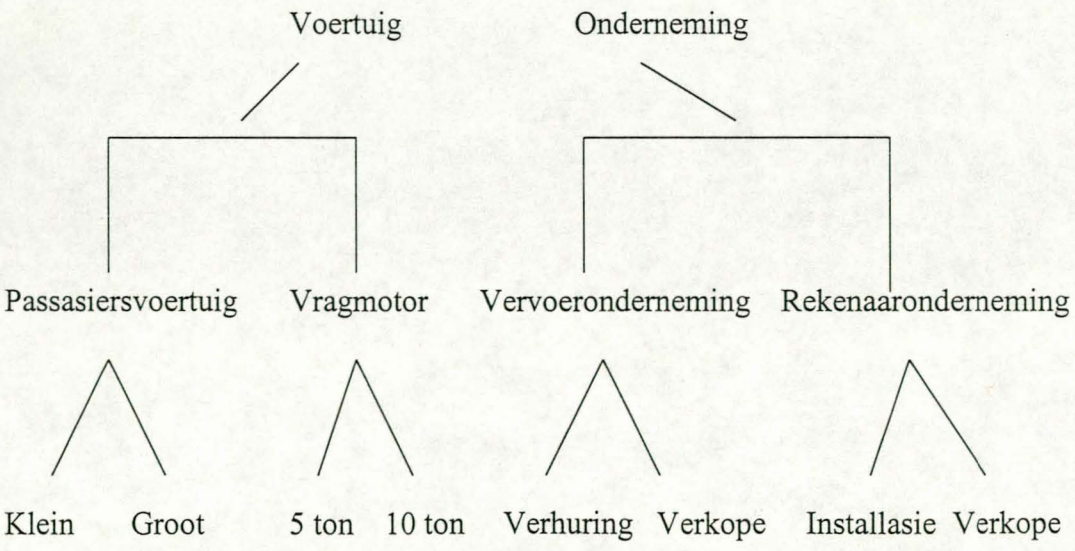
Daar bestaan hoofsaaklik drie tipes objekklasse, naamlik intervlak-, beheer- en entiteitobjekte (Jacobson; 1995: 114). Intervlakobjekte bestaan uit 'n groep aktiwiteite wat elk deur dieselfde bron uitgevoer word, met die doel om te

kommunikeer met die omgewing binne die onderneming. 'n Verkoopsagent is 'n voorbeeld hiervan, aangesien die persoon normaalweg sy werksfunksies uitvoer deur kommunikasie met ander rolspelers. Beheerobjekte bestaan ook uit 'n groep aktiwiteite vir dieselfde bron, maar die werkstake word nie uitgevoer deur interaksie met die ondernemingsomgewing nie. Beheeraksies verteenwoordig onder andere spesialis werksfunksies, is aktief en vervul rolle soos dié van produksiebestuurders en produkontwikkelaars. Entiteitobjekte is die hanteerbare items in 'n besigheid wat deur 'n reeks gebruike in die besigheid hanteer kan word, soos produkte en bestellings.

'n Alternatiewe tegniek vir oombliklike aktivering is die gebruik van prototipe-objekte. Hierdie objekte beskik elk oor hulle eie eienskappe, en word gebruik in modelle waar ooreenkomste tussen objekklasse nie algemeen is nie, maar waar objekte as unieke eenhede hanteer word.

'n Voorbeeld van 'n klassehiërargie word in Figuur 2.3 getoon. Hier is dit duidelik dat die voertuig en onderneming bo-aan die hiërargie is, met onderskeidelik die passasiersvoertuig en vragmotor ingedeel by die voertuig as subklas. Dit beeld 'n tipiese klassehiërargie uit, waar die ontwerper van die stelsel eerstens die voertuigklas spesifiseer, en daarna die onderliggende klasse daaruit sal aflei.

Figuur 2.3 'n Klassehiërargie in 'n vervoeronderneming



2.4.2 Oorerflikheid

Alhoewel oorerflikheid in verskeie bekende vorme bestaan, word dit hier beskou as 'n meganisme vir die toepassing van hergebruik van objekte, wat wissel tussen *klasoorerflikheid* en *dinamiese oorerflikheid*. Klasoorerflikheid word beskou as die enkele eienskap wat objek-georiënteerde programmering skei van ander programmeringsmetodes. Hierdeur is ontwerpers van groot programme in staat om die ooreenstemmende eienskappe van objekte in 'n model te gebruik om die model te vereenvoudig. Enkelvoudige oorerflikheid bestaan waar 'n objek beskik oor slegs een klas *ouer*, maar vir meervoudige oorerflikheid word meer as een klas ouer ingespan om eienskappe aan die ontwikkelde objek oor te dra (Kim & Lochovsky; 1989: 7).

Alle programmeringstale beskik nie oor die vermoë om meervoudige oorerflikheid aan te wend nie, wat meer komplekse modellering grootliks beperk. Dit is belangrik om wel die eienskap van meervoudige oorerflikheid te omseil, en vir reeds gevormde objekte nuwe eienskappe te programmeer. Vir die keuse tussen die ooreenstemmende metodes van geërfde objekte, moet die ontwerper hier besluit watter metode die mees toepaslike uitslag lewer. In gevalle waar geen uitsluitsel hieroor gegee word nie, behoort die program aan die hand van vaste reëls 'n keuse te maak tussen die metodes, of dit te kombineer.

Verskeie probleme ontstaan by die aanwesigheid van meervoudige klasse. Versteekte eienskappe wat deur oorerflikheid oorgedra word op die laagste klas, kan in geval van veranderinge aan die hiërargie van 'n model probleme veroorsaak. Wanneer eienskappe van die ouer klas aangepas word, beïnvloed dit outomaties die laer klasse, wat nie altyd die gewenste uitwerking het nie. Hierdie probleem staan bekend as *skema-evolusie* in objek-georiënteerde databasisse, en kom voor waar daar onsekerheid bestaan oor die behoud al dan nie, van die eienskappe van 'n verwante klas objekte.

Oorerflikheid het 'n ander rang binne die konteks van verteenwoordiging van kennis. Hier is dit die laer objekklasse wat kennis verteenwoordig, en gevalle van subgroepe beskik nie net oor al die eienskappe van die oorspronklike objek nie, maar word selfs

meer spesifiek geprogrammeer. Dit lei tot die beskouing dat subgroepe meer gespesialiseerd as die oorspronklike objekte is (Kim & Lochovsky; 1989: 8).

Klasoorerflikheid kan aangebied word as gedeeltelike oorerflikheid, waar slegs die gespesifiseerde eienskappe van die oorspronklike objek na die ontwikkelde objek oorgedra word. Die meganisme kan klashiërargie geweldig kompliseer, maar indien dit korrek toegepas word, bly die objekte in die regte verhouding tot mekaar verbind, en die ontwerper kan, deur enkele aanpassings, die model deurgaans opdateer. Die taal C++ maak voorsiening vir gedeeltelike oorerflikheid (Kim & Lochovsky; 1989:8).

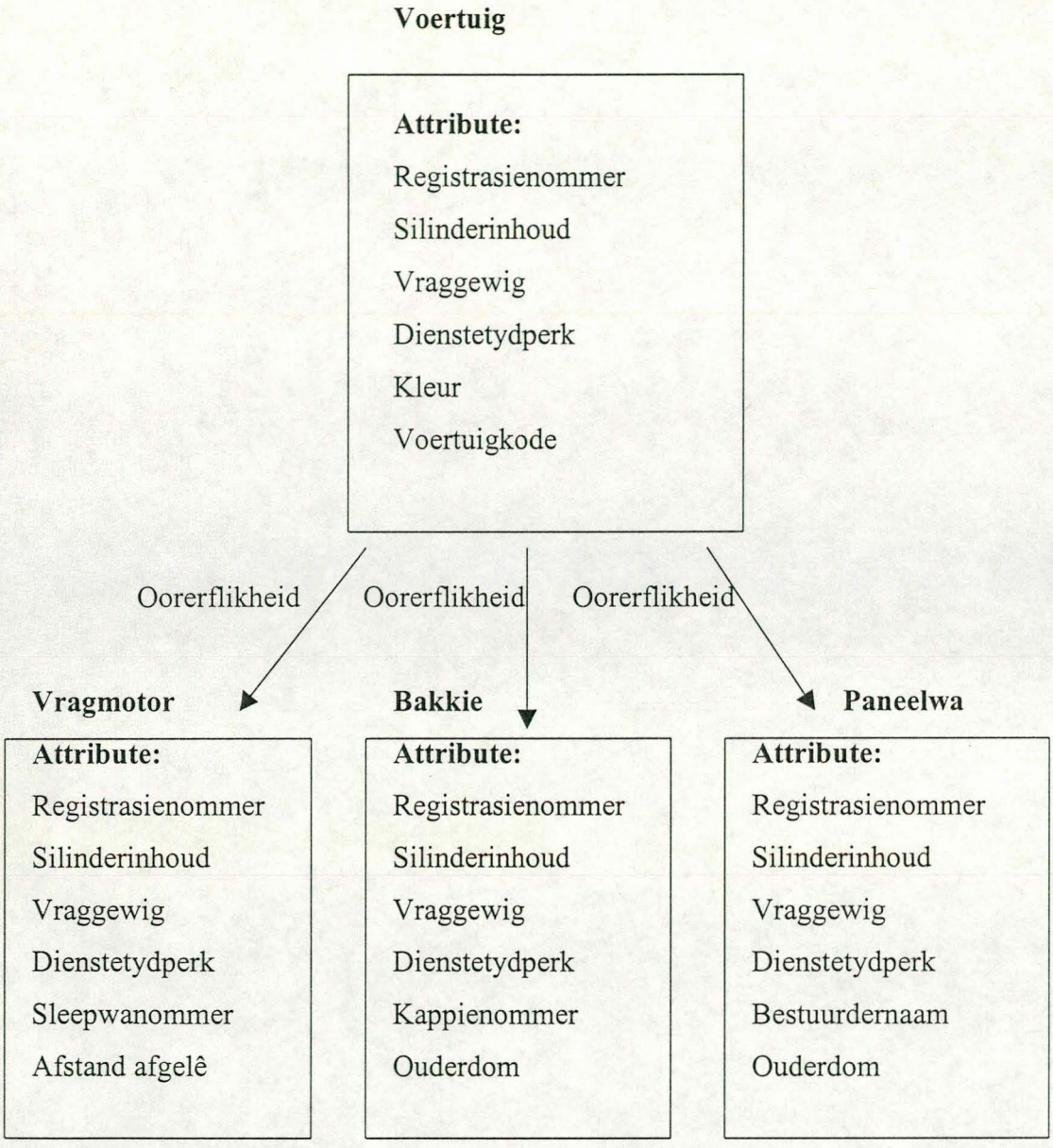
Die veranderinge wat aan klasse aangebring is, word nie beskou as dinamiese oorerflikheid nie, aangesien die definiëring van klasse nie 'n objekverwante aktiwiteit is nie. Dinamiese oorerflikheid tree in by die meganismes wat objekte toelaat om hulle gedrag tydens normale interaksie te verander. Hierteenoor staan skemaevolusie, wat ook in die objekmodel voorkom. Dinamiese oorerflikheid bestaan uit onderdeeloorerflikheid en omgewingoorerflikheid. Eersgenoemde vorm kom voor waar 'n objek 'n nuwe onderdeel of bewegende eenheid van 'n ander objek ontvang, teenoor omgewingsveranderinge wat in laasgenoemde geval die oorerflikheid aanwakker. Onderdeeloorerflikheid is niks anders as 'n uitruiling van waardes tussen objekte nie. Dit kan gesien word wanneer die objekklas veranderinge in 'n *oombliksveranderlike* beperk, selfs wanneer dit deur 'n objek gemanipuleer word.

Omgewingoorerflikheid is meer algemeen. Hier word 'n objek se gedrag bepaal deur die omgewing waarin dit geplaas word. Veranderinge in die omgewing beïnvloed noodwendig die gedrag van die objek, soos wanneer 'n paragraaf in 'n teksgedeelte die lettertipe, formaat, en ander karakteristieke aanneem. Dieselfde paragraaf neem nuwe karakteristieke aan wanneer dit na byvoorbeeld 'n voetnota of ander dokument oorgeplaas word. 'n Objek beskik normaalweg oor oombliksveranderlike en eie metodes, maar *delegeer* ook sommige boodskappe na verwante objekte, wat as prototipes bekend staan. Onderdeeloorerflikheid vind plaas wanneer 'n objek verwante objekte vervang, terwyl omgewingoorerflikheid plaasvind wanneer 'n objek sy eienskappe en dus ook dié van die verwante objekte verander. Wanneer die

simuleerder prototipes wil aanpas, word dit bloot in die vlak van die prototipe gedoen, sonder dat enige oorerflike effek beïnvloed word.

Figuur 2.4 dui die oorerflike verwantskappe tussen klasse skematies aan. Hier word die voorbeeld van voertuie in die onderneming gebruik. Verskillende voertuie word geëien met behulp van ooreenstemmende attribute soos gewig, ouderdom, brandstoftipe, diensrekords, en verskeie ander eienskappe. Hier word getoon dat die attribute ooreenstem en verskil, afhangende van die klas van die objek.

Figuur 2.4 Verwantskappe vir oorerflikheid



2.4.3 Polimorfisme

Die *polimorfistiese* funksie is een wat uniform op 'n verskeidenheid objekte toegepas kan word (Kim & Lochovsky; 1989: 10). Hiervolgens kan die programmeerder dieselfde funksie gebruik om byvoorbeeld *komplekse* getalle by *integrale* te voeg.

Klasoorerflikheid en polimorfismes is nou geskakel, aangesien dieselfde aktiwiteite wat aan die ouer-objek oorgedra word, ook op die kind-objek oorgedra kan word. Ondersteuning vir die polimorfisme is ook moontlik sonder die aanwesigheid van oorerflikheid. In *Unix* kan byvoorbeeld die aktiwiteite van *lees* en *skryf*, wat as *stroomaktiwiteite* bekend staan, geaktiveer word, ten einde al die objekte te beheer. Polimorfismes verbeter die hergebruik van programmatuur, aangesien dit nou moontlik is om nie net die huidige objekte te manipuleer nie, maar ook die objekte wat later gevorm word in *generiese* programmatuur.

2.4.4 Generiese klasse

Die meganisme van klasoorerflikheid word gebruik vir die skepping van algemene eienskappe by die hergebruik van objekte. Generiese objekklasse bereik dieselfde doelwit deur die gedeeltelike beskrywing van 'n klas en die vorming van parameters vir die onbekende. Daar bestaan twee kategorieë, naamlik *homogene* en *gereedskapobjekte*. Die homogene kategorie beskryf die ouer-tipe objekte wat alle soorte objekte kan hanteer, terwyl gereedskap-tipe objekte dié objekte verteenwoordig wat slegs 'n sekere tipe objek kan hanteer.

Vir gereedskapobjekte word die parameter verbind om die ouer-klas van die parameter aan te dui. Dit word vereis aangesien sekere objekte soos die *Lys* enige tipe objek kan insluit, terwyl gereedskapobjekte soos die *Sorteerder* aan die bogenoemde beperking onderhewig is. Die Sorteerder kan vanuit die Lys objekte neem, waarvan almal nie hanteer kan word nie. Alle objekte wat aan die beperkings voldoen, word dan in die Sorteerder uniform volgens die polimorfisme hanteer.

2.5 Objektipes

Die verskil tussen objekklasse en tipes lê daarin dat dié tipe klassifikasie die programmeerder in staat stel om die veranderinge aan die objekte staties te verifieer. Sodoende word die objekte outomaties teen onverwagse boodskappe beskerm. Die

programmeerder ken byvoorbeeld verskillende statiese vorms aan objektipes toe, sodat elkeen maklik uitkenbaar bly tydens programmering.

Tradisioneel is die toetsing vir 'n tipe-objek gedoen deur die vereiste dat die tipe uitdrukking vir aktiwiteite moet ooreenstem met die verwagte tipe. In polimorfistiese operasies en dinamiese bindings kan soortgelyke objektipes wel voorkom, of sekere tipes kan in andere ingesluit word. Een tipe objek konformeer tot die tweede tipe wanneer een gedeelte van die intervlak identies aan dié van die tweede objek is. Die eerste objek staan dus bekend as 'n subtipe van die tweede. Objekte is ekwivalent as beide tot die ander konformeer. Die intervlakke van objekte van verskillende tale kan verskil, maar bestaan normaalweg uit aktiwiteitsname, tipe argument, en invoerwaardes.

Die verskille tussen objektipes en -klasse kan duideliker wees wanneer objektipes bloot as spesifikasies beskou word. Dit is in die aanwesigheid van 'n dinamiese binding feitlik onmoontlik om die klas van die veranderlike staties te bepaal, maar 'n *tipe-toetsing* is wel moontlik. Gestel vergelyking 2.1 geld;

$$x = y + z \qquad 2.1$$

dan word die toets uitgevoer deur te bepaal of die vergelykingtipe korrek is. Beskou eerstens y, en bepaal of dit die operateur "+" ondersteun. Indien dit wel die geval is, word bepaal of z geldig is vir die argument. Dan sal die tipe inligting wat geldig is vir die argument, verteenwoordig word. Die tipe-inligting van y sal dus die tipe-inligting van $y + z$ verteenwoordig. Wanneer hierdie tipe inligting konformeer tot die tipe x (as $x + z$ ten minste die intervlak van x ondersteun), is die uitdrukking korrek volgens die tipe-toets.

Tipe-inligting kan uiters nuttig wees in generiese objekklasse. So byvoorbeeld kan die generiese Sorteerderobjek slegs die *TotallyOrdered* klasse sorteer. Hierdie beperking is 'n tipe-beperking, aangesien die model nie die klas ondersoek nie, maar bloot nagaan of 'n totale bestelling gedefinieer is.

In die meeste OG tale word die meganismes van *Empathy* en *Templates* gebruik, en die tale word beskryf volgens die interaksie tussen hierdie twee meganismes. Waar 'n objek 'n attribuut *leen*, staan dit bekend as Empathy. 'n Templaat bestaan uit die versameling van alle metodes en veranderlikes wat nodig is om 'n nuwe objek te definieer. In die geval waar 'n nuutgevormde objek geen attribute mag bykry of verloor nie, is die templaat *streng*. By streng template word die uniforme en onafhanklike aard van 'n klas verseker.

2.6 Samelopende lyne

Produksielyne wat gelyktydig funksioneer, staan bekend as samelopende lyne of parallelle lyne. Produksielyne wat op hierdie wyse in die stelsel geskakel is, moet met ander tegnieke hanteer word as wat die geval is met produksielyne in serie.

2.6.1 Bekende tegnieke van kommunikasie

Volgens Kim & Lochovsky het programmeringstale in die verlede oor twee tegnieke beskik om samelopende lyne en hulle kommunikasie te hanteer, naamlik:

1. Prosesse kommunikeer met behulp van onderling gedeelde passiewe objekte; en
2. Prosesse kommunikeer deur direkte boodskapversending.

Vir die eerste benadering sou die programmeerder tipies 'n groep passiewe objekte gebruik om die gedeelde geheue te struktureer, en dit te beskou as 'n soort aktiewe prosesobjek. 'n Meganisme word hiervoor benodig om te verseker dat die passiewe objekte in 'n gesinchroniseerde tydvak toegang tot die geheue verkry. In die tradisionele tale bestaan daar 'n aantal tegnieke wat as *slotte* beskryf kan word, wat bloot toegang tot die geheue beheer en sinchroniseer.

Die eerste benadering word deur die volgende nadele gekenmerk, wat die tweede benadering bevoordeel, naamlik:

1. Die model is nie-homogeen met aktiewe en passiewe objekte teenwoordig;
2. Twee aktiewe objekte kan slegs met 'n passiewe skakelobjek kommunikeer; en

3. Modelle moet beskik oor verskuilde boodskapoordragte vir toepassing op verspreidingsomgewings.

In die tweede benadering kan enige objek kommunikeer met enige ander objek, onafhanklik van die aktiewe of passiewe aard daarvan. Hier kan objekte geaktiveer word deur die oordrag van 'n boodskap, teenoor die prosesgedrewe aktivering van die eerste benadering. Boonop is sinchronisasie van aktiwiteite ingebou in die boodskapoordrag. Die aard van die boodskapoordrag moet egter gekies word, wat die modelreaksie beheer volgens gesinchroniseerde, *gedempte*, *eenrigting gefiltreerde*, of *terugkerende* maatstawwe. Die programmeerder kan hierdie tegniek gebruik om die reaksies in die model te manipuleer vir 'n meer getroue voorstelling van die werklikheid. So byvoorbeeld kan *buffers* en *stoot-* en *trekstelsels* in 'n hanteringsfabriek korrek nageboots word vir akkurater resultate.

2.6.2 Argumente vir samelopende lyne

Daar bestaan vier primêre argumente vir samelopende lyne, wat hier bespreek word (Kim & Lochovsky; 1989: 88), naamlik:

1. interaktiewe primitiewe,
2. hulpbronbestuur,
3. omhulselvorming, en
4. klasreëlmatigheid.

Interaksie tussen twee objekte vereis dat daar ten minste vanaf een objek 'n boodskap na 'n ander gestuur word, wat dit ontvang. Programmeringstale beskik dikwels oor meer as een tegniek van boodskapversending (Kim & Lochovsky; 1989: 80), wat elk gesinchroniseerd of ongesinchroniseerd is, in die interaktiewe primitiewe veld. Gesinchroniseerde boodskappe *blokkeer* die ingang van die objek totdat 'n bepaalde boodskap-verwante aktiwiteit plaasvind, terwyl ongesinchroniseerde boodskappe die objek toelaat om te produseer wanneer die tegniek lokaal uitgevoer is. Daar bestaan dus vier moontlike toestande vir die objek, naamlik *blokkeer versend*, *nie-blokkeer versend*, *blokkeer ontvangs* en *nie-blokkeer ontvangs*.

Vir hulpbronbestuur is veranderlikes verdeel of onverdeel, die *korrelgrootte* is groot of klein, en liggings kan afhanklik of onafhanklik wees. OG tale neem aan dat veranderlikes onverdeel is, tensy anders vermeld. *Korrelgrootte* verwys na die affiniteit wat tale vir objekte het met 'n ander korrelgrootte, normaalweg 'n groter affiniteit vir groter korrels. *Eenheidsvorming* verwys na die ideale geval in die OG model waar objekte elk hulle eie lokale toestand behou, en geen toegang daartoe moontlik is behalwe deur die lokale metodes nie. Klasreëlmatigheid is van toepassing waar *objekmigrasie* benodig word vir die balansering van lading. Dit is ook toepaslik waar objekte elk na dieselfde ligging verplaas word ten einde die koste van interaksie te verminder.

2.7 Tipes Objek-georiënteerde stelsels

In hierdie afdeling word twee tipes objek-georiënteerde stelsels bespreek, waarvan die eerste ondersteuning bied aan objek-georiënteerde toepassings, en die tweede 'n omgewing skep vir objek-georiënteerde programmatuurontwikkeling.

2.7.1 Objekhantering

Looptydondersteuning vir die model geskied deur verskeie aksies uit te voer (Kim & Lochovsky; 1989: 15), soos:

- benoeming,
- behoud,
- samelopende lynvorming,
- verspreiding,
- sekuriteit, en
- beheer oor vorming.

Die mate waartoe ondersteuning benodig word hang af van die toepaslike gebied van ondersoek, wat kan wissel van die enkelgebruiker met enkelvlaktoepassings tot die meervoudige gebruiker van verspreide, gelyklopende modelle. In beide gevalle word objekte geplaas in lokale en private, of verspreide en gedeelde omgewings.

'n Nadeel by die hantering van 'n model met behulp van objekte in die databasis, is dat daar nie 'n standaard bestaan vir die datamodelle nie. Algebraïese verwantskappe as ekwivalent van die objek-georiënteerde model bestaan nie, wat die afwesigheid van riglyne in modelontwikkeling veroorsaak. Die tekort aan standaardmeganismes vir die modelle en die onsekerheid van die tipe programmeringstale verklaar hierdie nadele. Die programmeerder word gevolglik met onsekerhede gelaat, wat uiteindelik daartoe lei dat modelle nie altyd met mekaar versoenbaar is nie.

2.7.2 Objek-geöriënteerde omgewings

Betekenisvolle kategorieë vir die ontwikkeling van stelsels is die gereedskap en die omgewing. Die werklike krag van hierdie programmering lê in die verminderde koste van modelontwikkeling, wanneer herhaalbaarheid en oorerflikheid korrek toegepas word. Gereedskap word benodig om objekte so te ontwerp dat bogenoemde eienskappe benut word, en om die veranderende omgewing van die databasis te bestuur.

Objekontwerp is 'n uitdaging vir die programmatuuringenieur. Hier moet die regte tipe objek binne die regte vlak klas geplaas word om die proses akkuraat en buigsaam te ontwerp. Die modelontwerp word gedoen met inagneming van die strategiese eienskappe van hergebruik, oorerflikheid, en aanpassingsmoontlikhede. Die uitdaging lê dus nie in die bou van 'n model nie, maar in die bou van 'n toepaslike, eenvoudige, en buigsame model. Daar bestaan geweldig baie objektipes en –klasse, wat die taak vir die ontwerper baie moeilik maak. Dit is dus raadsaam om slegs op sekere gedeeltes van 'n model te konsentreer, om dit vir 'n persoon moontlik te maak om die geheelbeeld van die program te begryp.

Versoenbaarheid tussen alle verwante stelsels is noodsaaklik vir doeltreffende elektroniese kommunikasie in die onderneming. Evolusie van die programmatuur veroorsaak egter instandhoudingsprobleme in koppelvlakke. Die ontwerper behoort dus te verseker dat interne kommunikasiekanale en koppelvlakke tussen objekte in stand gehou word. Dit is verder noodsaaklik dat die aanskaffers van hierdie

programmatuur duidelike vereistes stel vir kommunikasie met ander stelsels, en dat die opgegradeerde stelsels geïntegreer word by die ander komponente van die stelsel.

2.7.3 Die Semantiese model

Ontwerpers van semantiese modelle verwys soms na hulle modelle as objek-georiënteerd, omdat dit meganismes verskaf vir die strukturering van komplekse objekte. Daar is egter 'n belangrike verskil tussen die objek-georiënteerde model en die semantiese model.

Die OG model poog om data so te groepeer dat dit in manipuleerbare formaat is. Semantiese modelle word gebou om data voor te stel in 'n meer verstaanbare en interpreteerbare vorm. Laasgenoemde tegnieke konsentreer op die ontwikkeling van komplekse tipes, met min verwysing na die gedragskodes. OG modelle gebruik 'n meer abstrakte benadering tot die tipe data wat in die metodes saamgestel is. Sodoende is 'n objek in beheer van sy eie gedrag, want hy kan boodskappe ontvang, en hiervolgens sy reaksies kies en uitvoer met die ooreenstemmende metodes.

Daar bestaan programmatuur wat die eienskappe van beide modeltipes akkommodeer. Gesamentlik voorsien hulle metodes waarvolgens strukturele en gedragsaspekte van komplekse modelle gemodelleer kan word, wat die beperkings in die ontwerpproses verminder.

2.8 Meganismes

Objekte binne die model kommunikeer met mekaar en met modelle in ander omgewings. Hierdie interaksie word aangetoon en gedefinieer met sekere meganismes, wat ontwikkel is om die interaksie tussen objekte duideliker voor te stel.

2.8.1 Akteurgebaseerde benaderings

Alle objekte in die akteurgebaseerde stelsel word as akteurs beskou (Kim & Lochovsky; 1989: 88). In hierdie model bestaan elke objek as indiwiduele en unieke akteur, wat onafhanklik van enige ander objek reageer. Alle boodskappe en resulterende aksies vind samelopend plaas met die boodskappe wat ontvang is, en daar is geen implisiete ordening van aksies nie.

Alle akteurs toon die unieke karaktereienskappe van *identiteit* en *huidige gedrag*. Sodra die ontwikkeling van 'n akteur behou hierdie objek sy oorspronklike identiteit, selfs al verander die gedrag daarvan oor 'n tydperk. Net so behou 'n akteur ook sy huidige gedrag, selfs *lewenslank* in uitsonderlike gevalle ("unserialized").

Vir die modellering van die vervoeronderneming wat van swaar voertuie en drywers gebruik maak, kies die ontwerper tipies die onderneming as 'n omgewing, met die drywers, vragmotors, en kantoorpersoneel as akteurs in die stelsel. Die akteurs se gedrag word deur hulle eie karaktereienskappe en die aksies van ander akteurs bepaal. So byvoorbeeld sou 'n drywer 'n vragmotor bestuur tot die vragmotor breek, en dit dan óf self herstel óf laat herstel. Hierdie bedryfsteurings word normaalweg deur 'n karaktereienskap van die vragmotor, of deur die hantering van die drywer veroorsaak, soos 'n ratpen wat breek as gevolg van 'n vermoeidheidsbreuk of drywersgewoontes.

Met behulp van die eienskappe van akteurs is dit moontlik vir die ontwerper om die tipiese gedrag van akteurs in 'n bedryf vas te vang en vir hergebruik te stoor. Die modelleringsproses word hierdeur vereenvoudig, dit vergemaklik die ontwerper se werk, en dit verlaag die ontwerpkoste.

2.8.2 Argitektuur

Tydens die ontwikkeling van die OG model van 'n onderneming moet die navorser die relevante strukture en verwantskappe visualiseer, in die proses wat as ondernemingsargitektuur bekend staan. *Argitektuur* verwys in hierdie verband na die

tipe ondernemingstruktuur en die verskillende tipes argitektuur wat die modelleringstaal toelaat. Hier word meer uitgebrei op die modelleringstaal se moontlike argitektuur.

Die argitektuur van 'n onderneming is hoofsaaklik die belangrikste statiese eienskappe daarvan, soos die funksionele departemente, afdelings, en sub-afdelings (Jacobson; 1995: 97). Menslike en ander hulpbronne, prosesse, en koppelvlakke tel ook onder hierdie argitektuur. Elemente wat gekoppel is, vorm 'n struktuur, beskik oor beperkings, lewer uitsette, en daar kan sekere waardes aan toegeken word.

Die doelwit van die argitektuur is nie net om die ondernemingsmodel voor te stel nie, maar ook om die verskillende reaksies van 'n onderneming op die verskeidenheid situasies te toon. Die groep reaksies van die onderneming op 'n kliënt se aksies vorm 'n *proses*. Die proses kan verdeel word in onderlinge prosesse, wat deur die model uitgebeeld word as geskakelde entiteite. Dit is 'n uitdaging om die onderlinge prosesse se verwantskap en dié van die oorhoofse proses korrek te modelleer. Hiervoor moet die modelleringstaal voorsiening maak, sodat die prosesse wat as objekte voorgestel word, kan deel vorm van die OG onderneming se ontwerp.

Die navorser moet verder die insette van die buitewêreld en die gevolglike vloei van produkte in die onderneming beskryf. Insette en uitsette moet kliëntgeoriënteer wees, aangesien dit die kommunikasie met die kliënt verbeter. Die ontwerpte model sal altyd onderhewig wees aan veranderinge, aangesien ondernemings gereeld aanpassings moet maak in hulle bestaande prosesse. Die argitektuur moet dus 'n buigsame raamwerk vorm, sodat die ontwerper die konstante veranderinge in die model kan aanbring sonder om die model te herontwerp.

Die model moet ook bestaan uit twee gedeeltes, naamlik die eksterne en die interne model (Jacobson; 1995: 100). Die eksterne model beskryf die onderneming en die eksterne koppelings daarvan. Dit beskryf ook die prosesse binne die onderneming waarin die kliënte en ander belanghebbendes mag belangstel. Die eksterne model staan as die *gebruiksgeval* bekend, aangesien dit die onderneming se prosesse oftewel gebruiksgevalle beskryf. Die interne model beskryf al die ondernemingsprosesse, deur volledig te verwys na die verskillende interne werkstake wat hulle uitvoer, en die

tipe hulpbronne benodig. Die interne model moet die toedelings aantoon van die werkstake aan funksies waarin 'n onderneming verdeel kan word. Dit dui dus aan waar 'n hulpbron vir 'n spesifieke taak opgespoor kan word.

Die interne model is objek-georiënteerd, en dit is nodig om verdere differensiasie toe te pas om die model duideliker te maak. Onderskeid word getref tussen objekte wat werkstake voorstel en objekte wat voorwerpe voorstel. Net so word werkstake ook verdeel in kliëntverwante take soos verkope, en interne werkstake soos produkbeplanning.

Die interne model kan *ideaal* of *reëel* wees (Jacobson; 1995: 101). Die ideale model neem nie die praktiese toepassing van die onderneming in ag nie, soos wanneer ondernemings geografies uitgebrei of verdeel is. Die reële model neem hierdie en ander faktore in ag, soos die feit dat werknemers nie die prestasievlak van die ideale model bereik nie. In die praktyk bestaan die interne model gewoonlik vir die ideale en soms ook in die reële geval.

Tydens modellering sou die ontwerper tipies bepaal waarvoor die ondernemingstelsel verantwoordelik is, en waarvoor die omgewing verantwoordelik is. Hieruit blyk duidelik dat die grens tussen die onderneming en die omgewing baie belangrik is. Normaalweg is dié grens onduidelik wanneer die modelleringstaak begin, maar gaandeweg word dit duideliker soos die model vorm aanneem.

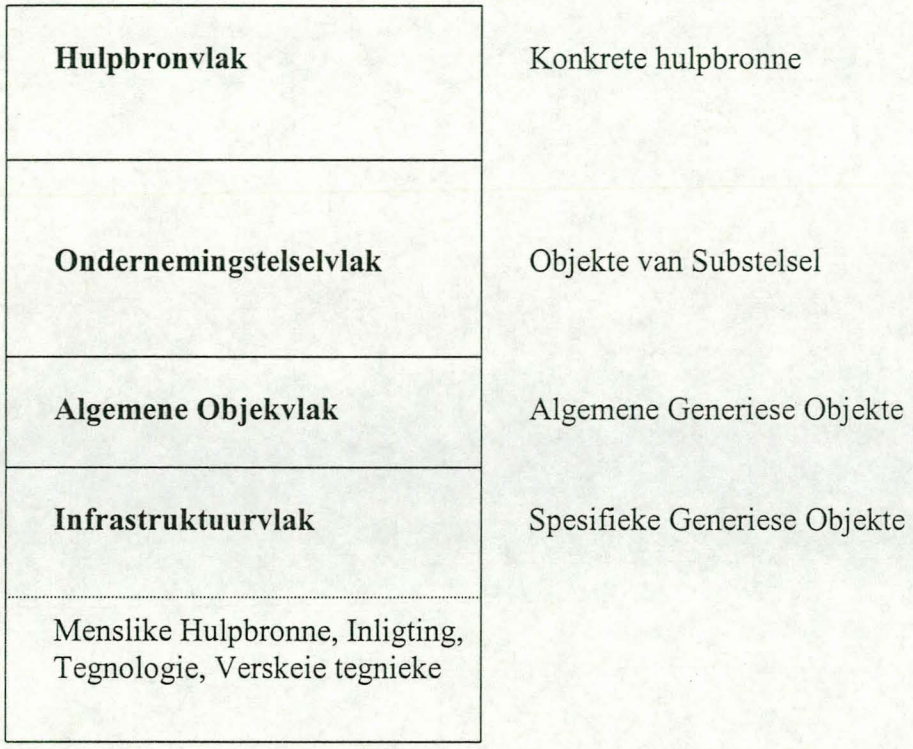
Die ondernemingsmodel word gevul met akteurs wat elkeen 'n rol binne die omgewing speel. Akteurs kan verdeel word in menslike en meganiese objekte.

Argitektuur vereis die korrekte benoeming van objekte, wat impliseer dat gebruikers in die aktiewe vorm benoem word. Die werkstasie-tipe objek voer dan 'n aksie uit op die gebruiker-tipe objek, soos die sortering van bewegende items wat deur die personeellede uitgevoer word.

Die vlakke in die argitektuur van 'n onderneming bestaan hoofsaaklik uit die ondernemingstelsel, die algemene objekvlak, die infrastruktuur, en die hulpbronvlak. Die laer vlakke verwys na die uitvoering en implementering van die model, terwyl die

hoër vlakke meer verwant is aan die onderneming self. Figuur 2.5 is 'n voorbeeld wat Jacobson (1995: 332) gebruik om die vlakke in die argitektuur te verduidelik.

Figuur 2.5 Vlakke in die ondernemingsargitektuur



Elke vlak in die argitektuur verteenwoordig 'n tipe funksie of spesialiteitsarea in die onderneming. Die vlakke is verwant aan mekaar, en kommunikeer deur middel van die boodskappe tussen objekte. In die vervoeronderneming sal die algemene objekte tipies die kliënte, rekeninge, takkantore, en stoorarea wees. Die objekte in die substelsels is aktiwiteite soos die opstel van kontrakte, uitvoer van aflewerings, en die betaal van rekeninge. Die algemene objekvlak is die area waarin die ondernemingsprosesse sigbaar is, en waarin verskeie ooreenstemmings tussen kontrakteurs kan voorkom.

Die raamwerk kan beskik oor twee belangrike klashiërargieë, naamlik *MessageContainer* en *ContainerRepresentative* (Islam; 1995: 60). Die MessageContainer is 'n klas wat die entiteit vorm waarin boodskappe met 'n buffer hanteer word. Prosesse kommunikeer dus deur boodskappe na houers te stuur, waarin objekte gestoor word. Die houer is dan die skerm wat bepaal hoe en wanneer elke objek die boodskap ontvang. Om hierdie boodskap te ontvang, moet die proses 'n

subklas van die MessageContainer ontwikkel en bind, met behulp van die *NameServer*. Die prosesse stuur dan boodskappe deur die *ContainerRepresentative* wat vanuit die NameServer verkry word. Die programmeerder kan boodskapgroottes reguleer deur die spesifieke eienskappe daarvan in die MessageContainer te programmeer, soos objekgrootte en objektipe.

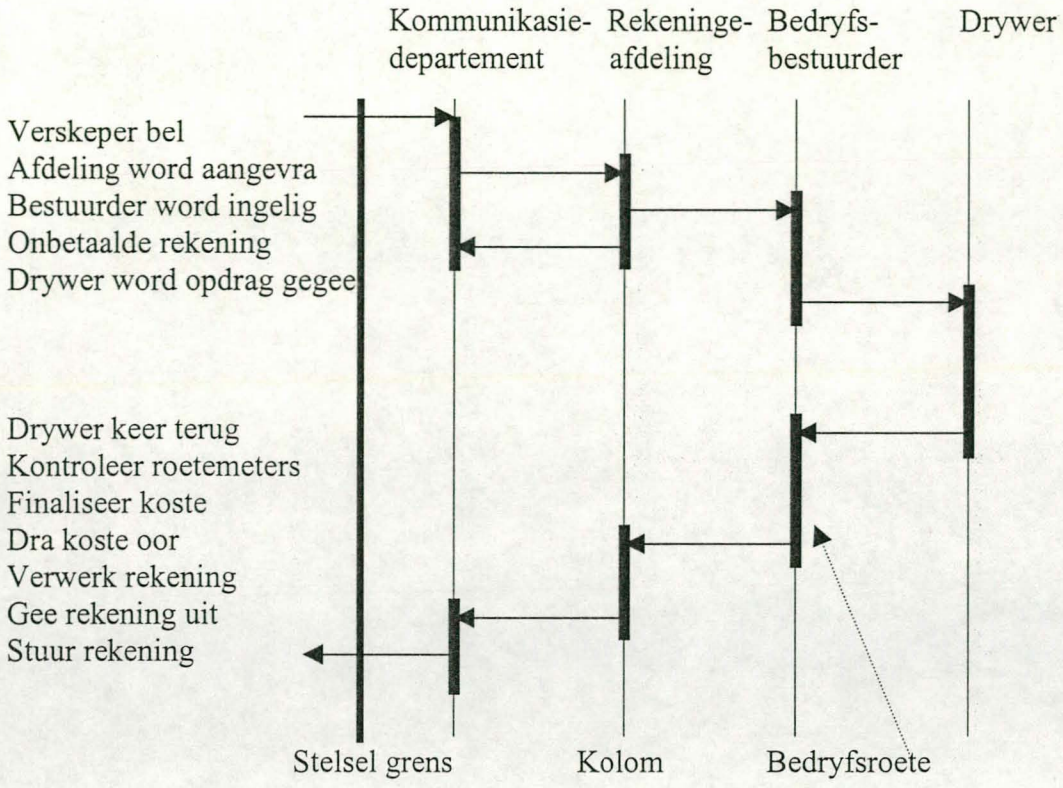
2.8.3 Die Interaksiediagram

Die roete wat items deur die model volg, word soms beïnvloed deur die samewerking van 'n aantal objekte, wat kommunikasie en interaksie meer kompleks maak (Jacobson; 1995: 132). Vir 'n duidelike uitbeelding van die interaksies wat betrokke is by sodanige proses, kan die ontwerper 'n interaksiediagram opstel. In die diagram stel die vertikale kolomme objekte voor. Die volgorde hiervan is nie belangrik nie, maar behoort so gekies te word dat die interaksie logies en eenvoudig vertoon word. Prosesgrense sluit die omgewing af, en kan ook verskillende omgewings verdeel. Die afstande tussen die gebeure het geen betekenis nie, aangesien opeenvolgende gebeure plaasvind deur middel van kommunikasie tussen objekte. In die linkerkantste kolom word die gedragsfunksies beskryf wat die objekte uitvoer, bekend as *prosespaaie*. Die prosespaaie word in die kolomme deur reghoeke voorgestel.

Interaksiediagramme word lank reeds in die telekommunikasiebedryf gebruik, veral vir die voorstelling van die interaksie tussen apparatuur. Jacobson het die konsepte van interaksie vir programmatuur van objek-modelle ontwerp in 1968, en vir OG programmering in 1987 verwerk.

Figuur 2.6 is 'n interaksiediagram vir 'n vervoerkontrakteur wat met swaarvoertuie vrag per pad vervoer. Hier kan gesien word hoe die roete lyk wat items deur die model volg, en watter aksies uitgevoer word om die items te vervoer.

Figuur 2.6 Interaksiediagram van sekere aktiwiteite in 'n vervoeronderneming



2.9 Die Ondersteuning van inligting

OG programmering gee aan die ontwerper die ondersteunende data vir die herontwerp van 'n stelsel (Jacobson; 1995: 237). Een van die belangrikste gereedskapstukke in hierdie proses is informasietegnologie (IT). Enige ondernemingsproses benodig data ter ondersteuning van sy aktiwiteite, wat die onderneming meer doeltreffend kan laat funksioneer indien dit korrek gebruik word. Die navorser behoort 'n inligtingstelsel, wat die prosesse ondersteun, reeds in die beginfase van die proses van herontwerp te gebruik. Sodoende kan hy die resultate van die nuwe prosesse evalueer, selfs voor die fisiese veranderinge aangebring word. IT moet deur ander stelsels ondersteun word vir 'n realistiese uitbeelding en ontwerp van stelsels en prosesse, soos geografiese verspreidingstelsels en eenvoudige kliënt-teenoor-dienspunt-tegnieke.

Objekoriëntasie-tegnologie bied nie 'n kitsoplossing nie, maar moet hanteer word as 'n basiese tegniek wat deur ander tegnieke ondersteun word. Sodanige tegniek is objek-georiënteerde programmatuuringenieurswese (OGPI). Programmatuur-

ontwikkeling is 'n ondernemingsfunksie wat hom daarop toespits om rekenaarprogramme vir die betrokke onderneming te ontwikkel ten einde die aktiwiteite van die onderneming meer vaartbelyn te maak. Akteurs in hierdie funksie is normaalweg die kliënte, wat die bepalers is van die vereistes vir die ondernemingsfunksies. Kliëntebehoeftes vorm die riglyne vir die ontwikkeling van programmatuur, aangesien die stelsel se hoofdoelwit is om eerstens die kliënte se behoeftes te bevredig.

Die belangrikste hanteerders van inligtingstelsels is die persone wat dit bestel en gebruik in die onderneming. Hierdie persone stel hoofsaaklik belang in wat die stelsel kan doen, en hoe hulle dit moet gebruik (Jacobson; 1995: 240). Ander hanteerders sluit in:

- Navorsers en ontwikkelaars van soortgelyke stelsels;
- Die inligtingprosesleier en –lid; en
- Die eienaar en die persone verantwoordelik vir die evaluasie.

Die belangrikste ontwerpsaspek van hierdie stelsels is om dit so te ontwerp dat funksionaliteite saam gegroepeer is. Dit word gedoen om te verseker dat die algehele stelsel nie beïnvloed sal word wanneer veranderinge aangebring word nie.

Die bou van inligtingstelsels is die bouproses van verskillende modelle wat elk verskillende aspekte van die stelsel beskryf. Daar is ses basiese aktiwiteite wat ontwikkelaars gewoonlik tydens die ontwikkeling van hierdie stelsels volg (Jacobson; 1995: 242).

1. *Bepaling van vereistes*: Die ontwikkelaar samel alle idees, voorstelle, vereistes, en beperkings deur die kliënte gestel, in.
2. *Analise van vereistes*: Hier word gekontroleer dat alle vereistes konsekwent toegepas, en niks uitgelaat word nie.
3. *Ideale ontwerp*: Die struktuur van die stelsel word robuus ontwerp waar veranderinge aangebring moet word.
4. *Reële ontwerp*: 'n Basismodel word ontwerp vir die implementering van die inligtingstelsel.
5. *Implementering*: Hier word die stelsel geïmplementeer.

6. *Toetsing*: Die nakoming van alle vereistes word hier getoets.

Voordele van die elektroniese databasis is legio. Aangesien baie ondernemings vandag van gevorderde programmatuur gebruik maak vir die hantering van hulle data, bestaan die moontlikheid dat 'n onderneming sy data met andere kan deel vir wedersydse voordele. Die koppelvlakke en data moet net so ontwerp word dat dit leesbaar is vir en versoenbaar is met beide stelsels. So byvoorbeeld kan vervoerkontrakteurs met hulle grootste kliënte inligting elektronies uitruil om byvoorbeeld die roetes en aankomstye vir vragmotors beskikbaar te stel. Sodoende verhoog dit die vlak van kommunikasie en verbeter dit die dienskwaliteit. In die geval van laat aflewerings kan die kontrakteur die kliënte op hoogte hou van die huidige situasie, en redes verskaf indien 'n vrag laat afgelewer word.

Dit is egter in OG modelle waar elektroniese inligtingmanipulasie die grootste voordeel bied. Nou kan groot databasisse doeltreffend bestuur word, en tussen rolspelers versprei word vir groter kliëntetevredenheid. Waar dit voor die koms van die persoonlike rekenaar baie moeilik was om groot databasisse te hanteer en te verwerk, kan dit nou goedkoper bestuur word tot voordeel van die kliënt en die kontrakteur.

Om te verseker dat kliënte tevrede sal wees met die produk, word die Beta toets uitgevoer. Die Beta *toetshanteerder* skep 'n Beta *produk*, wat bestaan uit 'n uitvoeringskode en 'n stel hulpmiddels. Die Beta produk word aan die Beta *kliënt* gelewer, en neem alle reaksies van die Beta kliënt op. Indien alle vereistes in die stelsel ingebou is, behoort die Beta toets geen probleme op te lewer nie. Hierdie toets is net 'n kontrole oor die mate waartoe die model aan die vereiste voldoen (Jacobson; 1995: 268).

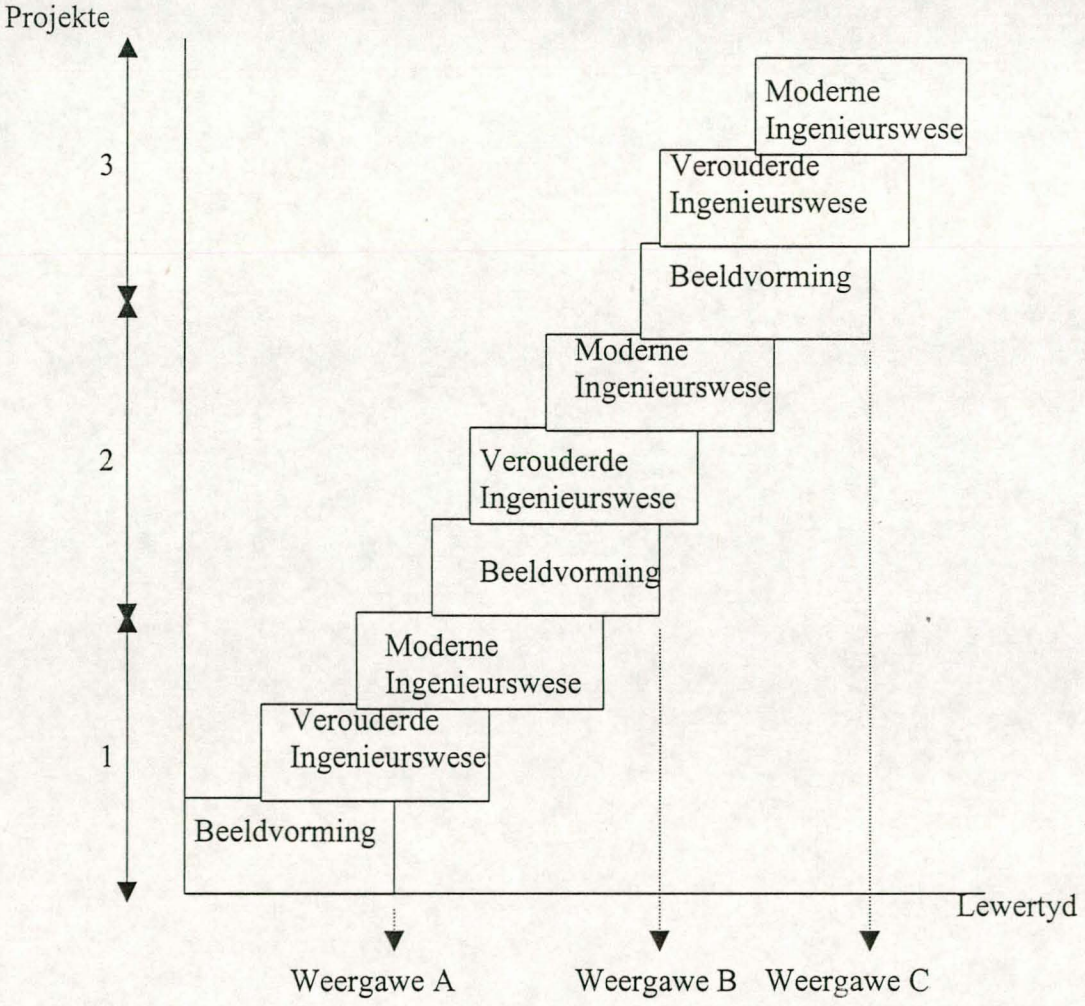
Stapsgewyse ontwikkeling van projekbeskouings word in Figuur 2.7 getoon. Wanneer die eerste projek begin word, kan 'n tweede projek ook 'n aanvang neem, sodat die twee gelyktydig ontwikkel (Jacobson; 1995: 304). 'n Nuwe weergawe van die objekspesifikasies word ontwerp soos die aktiwiteite afgehandel word. Objekspesifikasie is die lys doelwitte wat deur die bestuur ontwikkel word vir die nywerheidsprosesse. Weergawe A en B in Figuur 2.7 verskil ten opsigte van die

afronding of uitbreiding van die oorspronklike, en ten opsigte van die byvoeging van nuwe doelwitte. Daar bestaan drie hoofredes vir die aanpassing van die objekspesifikasies, naamlik:

1. Die doelwitte is nie duidelik genoeg nie;
2. Die projekspan besef dat die tyd en hulpbronne nie voldoende is om al die doelwitte te bevredig nie; en
3. Sommige doelwitte blyk onrealisties vir implementering te wees.

Nuwe doelwitte kan die projek se koste verhoog en die tydsduur verleng. Die toevoegings raak ook toenemend duurder met die vordering van die projek. Ontwerpers moet dus waak teen die herontwerp en spesifisering van modeldoelwitte as hulle nie die implikasies deeglik nagevors en oorweeg het nie.

Figuur 2.7 Iterasies en inkremente word hanteer met opgedateerde resultate



2.10 Verspreide objekte

'n Ander benadering vir OG programmering is om die modellering te beskou as die proses waarvolgens gevormde objekte in 'n korrekte verspreidingsnetwerk geplaas word. Volgens Islam (1996: xi) word die rekenaarindustrie tans gedryf deur hoofsaaklik twee doelwitte, naamlik verspreide toepassings en massavervaardiging. Die onlangse neiging in die industrie toon 'n verhoogde belangstelling in die veld van verspreide rekenaartoepassings, wat duidelik gesien kan word in die groeiende getalle ondernemings wat hulle kernbesigheid hiermee hanteer. Tegnologieë soos *Sun Microsystems* se *Java*-sisteem tesame met die COBRA en OLE/COM standaarde verseker dat die rekenaar as die netwerk beskou kan word. Konsepte soos outonome en intelligente *agente* wat netwerke vir hulpbronne ondersoek, het van teoretiese projekte na praktiese toepassings ontwikkel.

Die neiging na doelgemaakte stelsels word hoofsaaklik deur twee faktore veroorsaak, naamlik aanvraag en ontwikkelingskoste. Die dae van vaste dienskettings of voorsieningskanale en onbuigsame vervaardiging of dienslewering, is volgens Islam (1996: xi) iets van die verlede. Om aan die nuwe tendense te voldoen, is dit belangrik dat 'n onderneming buigsaam moet wees, maar ook 'n geweldig hoë reaksiesnelheid moet kan handhaaf. Dit vereis nie net 'n doeltreffende elektroniese ondersteuningstelsel nie, maar ook 'n stelsel wat aanpasbaar is, en die ondernemingsfunksies deeglik opsom. Die kombinasie van doeltreffende toepassings met doelgemaakte eienskappe bly steeds 'n kuns om te ontwikkel.

2.11 Algemene probleme

Soos met die meeste ander gonswoorde het daar ook sekere wanopvattings oor OG programmering ontstaan. Dittrich (1991: 2) beweer dat die konsep soms oorskat, opgehemel, en selfs oorvereenvoudig word. Tipiese simptome van hierdie verskynsels is:

- Ondernemings soek volwasse objekte waaroor hulle reeds beskik, maar waarvan hulle nie die betekenis verstaan nie;
- Ondernemings beweer dat hulle oor OG modelle beskik of daarmee werk; en

- Bekende en ou tegnieke verskyn weer onder 'n nuwe naam.

Volgens Dittrich (1991: 4) duur dit normaalweg 15 jaar vandat die eerste idees van 'n nuwe tegniek gepubliseer word tot dit algemeen gebruik word in programmatuurtoepassings. Aangesien OG programmering eers sedert 1987 ontwikkel word vir gevorderde rekenaartoepassings, is die gebruik daarvan nog ietwat beperk.

Die meeste tegnieke vir die hantering van databasisstelsels is gebaseer op 'n onbubbelsinnige en helder basis, wat nie die geval met OG programmering is nie. Hier is die aanvanklike reëls en beginsels vaag, met toepassings wat van verskillende grondbeginsels gebruik maak. Gevolglik is dit moeilik om selfs 'n norm te vind vir die teoretiese beginsels, en die verskillende konvensies veroorsaak soms aanpasbaarheidsprobleme waar stelsels met mekaar moet kommunikeer. Die meer algemene gebruik van rekenaartoepassings in die veld het die eenwording van die belangrikste konvensies baie bevoordeel, en vandag is dit wel moontlik vir die meeste ontwerpers om stelsels te integreer.

2.12 Objek-geöriënteerde toepassings

Die vorming van homogene eenhede wat op mekaar reageer, het sekere implikasies. Die eenhede wat in hierdie proses as objekte bekend is, moet normaalweg met mekaar kommunikeer, en elk is tot 'n sekere voorafgeprogrammeerde mate aan mekaar verwant. So 'n netwerk objekte beskik oor 'n karakter wat ontstaan uit die strukturele parameters daarvan. Parameters gee 'n goeie aanduiding van die verwantskappe tussen die objekte en die algemene vorm van die netwerk. Hierdie eienskappe staan bekend as *strukturele karakteristieke*, *strukturele eienskappe*, *strukturele waardes* of bloot as die *struktuur* (Zeggelink; 1993).

Die modellering van menslike gedrag en dié van 'n produksiestelsel toon baie ooreenkomste, gebruik soortgelyke gereedskap, en word beskryf met konsepte van dieselfde betekenis. Dit verklaar ook die ooreenstemmende terme en begrippe.

Daar bestaan teorieë oor die redes waarom persone vriende nodig het, wat onder andere in 1986 deur Miell & Duck gevorm is. Net so moet objekte om verskeie redes verbind wees aan ander objekte om die netwerk te voltooi. Objekverwantskappe moet geskep, onderhou, aangepas, of getermineer word, net soos vriendskappe. Verwantskappe tussen persone in 'n vriendskapsverhouding toon verskillende eienskappe in die teenwoordigheid van andere, wat ook in die geval van OG programmering plaasvind. Dikwels word verwantskappe tussen objekte met grafieke of diagramme aangetoon, en dieselfde grafieke of diagramme kan ook gebruik word om menslike vriendskapsverwantskappe aan te toon.

Volgens Zeggelink stel Davis in 1963 die aanname in sy statiese balansmodel dat die waarskynlikheid vir twee indiwidue om van mekaar te hou, groter is indien hulle in sommige opsigte ooreenstem. In die dinamiese balansmodel word sekere verwantskappe wat wanbalans veroorsaak, weggelaat om groter ewewig te bereik. Statiese en dinamiese modelle kom ook voor in produksieprosesse, waar objekte sekere aanpassings aan die stelsel tydens of na afloop van 'n proses maak. Objekte van ooreenstemmende klasse kommunikeer ook op 'n hoër vlak as dié van verskillende klasse. Die ooreenstemming tussen objekte bepaal in hierdie geval die aangetrokkenheid tussen hulle.

3. Simulasie van die hanteringsfunksie

Simulasie is 'n baie kragtige stuk gereedskap wat vir 'n groot aantal toepassings gebruik kan word, en wat veral van toepassing is op komplekse stelsels. As gevolg van die kompleksiteit van sekere stogastiese modelle is die tegniek van simulasie soms die enigste betroubare tegniek om prosesse te modelleer. Vereenvoudiging kan soms die betroubaarheid van 'n tradisionele model benadeel, en simulasie bied hier 'n goeie oplossing.

3.1 Agtergrond

Simulasie word gedefinieer as die tegniek wat 'n werklike proses namaak, normaalweg binne 'n korter berekeningstydperk as die werklike proses (Winston; 1994: 1180). Volgens Chase en Aquilano (1992: 800) is simulasie die nabootsing van die funksionering van een stelsel met behulp van 'n ander een. Die simulasiemodel neem normaalweg die vorm van 'n reeks aannames oor 'n proses aan, uitgedruk as wiskundige of logiese verwantskappe tussen die betekenisvolle objekte in 'n stelsel. Normaalweg word 'n rekenaar gebruik vir die generering van verteenwoordigende steekproewe of maatstawwe van prestasie. Simulasie kan dus gesien word as 'n eksperiment van die werklikheid, waar die resultate verteenwoordigende steekproefelemente is.

Om 'n model voor te stel, kan die tegniek een van twee prosesse uitvoer. Simulasie voltooi 'n vooraf gespesifiseerde aantal iterasies om 'n kontinue model voor te stel, of word geprogrammeer om oor 'n tydperk aktief berekeninge uit te voer.

3.2 Voor- en Nadele

Die belangrikste voordeel van simulasie is die logika daarvan, wat veroorsaak dat dit makliker toepasbaar as sommige analitiese tegnieke is. Analitiese tegnieke vereis soms vereenvoudigende aannames, terwyl simulasietegnieke weinig sodanige

beperkings het. Groter buigsaamheid is moontlik met simulasietegnieke, aangesien een model herhaaldelik gebruik kan word om 'n verskeidenheid beleide, parameters en ontwerpe te analiseer. 'n Maatskappy kan byvoorbeeld verskeie voorraadtegnieke toets, en die beste een implementeer, sonder om vooraf enige fisiese veranderinge aan te bring.

Simulasie is egter nie 'n optimeringstegniek nie, maar word normaalweg gebruik om *wat as*-vrae te beantwoord. Dit beteken dat simulasie vir veranderinge aan byvoorbeeld 'n produksielyn, antwoorde kan bied ten opsigte van die onsekere veranderde uitsette, binne perke van akkuraatheid. Optimering met simulasietegnieke is wel moontlik, maar dit is 'n stadige proses. Simulasie is ook 'n duur tegniek, maar raak meer bekostigbaar met doelgemaakte programmatuur, verhoogde programmeringspoed, en verbeterings in sekere simulasiemetodologieë.

Selfs by die opleiding van bestuurders kan simulasie 'n baie handige instrument wees. Dit kan gebruik word om die werklike stelsel voor te stel, om aan te toon watter invloed veranderinge op die stelsel het, en om nuwe teorieë te ontwikkel vir wiskundige en organisatoriese verwantskappe.

Die lys voordele, dikwels in vergelyking met wiskundige modelle, is soos volg (Chase en Aquilano; 1992: 816):

1. Modellering lei gereeld tot 'n beter begrip van die probleem.
2. Tyd kan in simulasie saamgepers word, wat jare se ervaring na enkele sekondes prosestyd kan verander.
3. Simulasie onderbreek nie aktiwiteite nie.
4. Simulasie is baie meer algemeen as wiskundige modelle, en kan soms gebruik word waar wiskundige modelle nie gebruik kan word nie.
5. Simulasie kan as spel vir opleidingservaring gebruik word.
6. Simulasie bied 'n meer realistiese nabootsing van die werklikheid as wiskundige modelle.
7. Onsekere toestande kan ook met behulp van simulasie benader word, maar nie wiskundig beskryf of bewys word nie.
8. Baie standaardprogramme is beskikbaar en bekostigbaar.
9. Simulasie beantwoord *wat as*-vrae.

Die lys nadele is soos volg:

1. Baie tyd en geld wat aan simulasie spandeer word, waarborg nie antwoorde op alle probleme nie.
2. Uitsette van simulasiemodelle kan nie bewys word of as heeltemaal foutloos bestempel word nie.
3. Simulasiemodelle kan vooraf een uur tot 100 werkersjare duur om te bou, wat dit duur en tydrowend kan maak.
4. Wiskundige modelle kan meer akkuraat wees as gevolg van die ewekansige element in simulasie.
5. Betekenisvolle hoeveelheid rekenaartyd kan in beslag geneem word.
6. Simulasietegnieke kort steeds 'n gestandaardiseerde benadering, wat beteken dat indiwidue twee verskillende modelle vir dieselfde proses kan bou.

Alle wiskundige modelle kan met behulp van simulasie ook uitgevoer word, maar simulasie is nie noodwendig die beste metode nie. Alhoewel wiskundige modelle meer akkuraat en goedkoper kan wees, het simulasie geen grense nie en weinig beperkings. Simulasieprogramme raak toenemend verbruikersvriendeliker en maak die bou van modelle al hoe makliker.

3.3 Modellering van 'n stelsel

In die meeste toepassings vir simulasie moet die navorser die een of ander stelsel simuleer. Hiervoor is 'n goeie begrip van die omvang van konsepte in die stelsel nodig.

3.3.1 Basiese terminologie

'n *Stelsel* is die versameling entiteite wat indiwidueel en gesamentlik reageer om een doel te bereik. Die samestelling en toestand van dinamiese stelsels kan oor 'n tydperk verander, en die meeste stelsels is dinamies. Die toestand van die stelsel verwys na die versameling veranderlikes wat die stelsel beskryf. 'n *Diskrete stelsel* se

veranderlikes kan net op 'n diskrete aantal tye verander, terwyl 'n *kontinue stelsel* se veranderlikes gedurig in die modelleringstydperk verander. 'n *Statiese simulasiemodel* is die voorstelling van 'n stelsel op 'n spesifieke tydstip, terwyl 'n dinamiese model 'n stelsel voorstel wat oor 'n tydperk varieer (Winston; 1994: 1184).

3.1.2 Probleemdefinisie

Doelwitstelling en die spesifisering van die onderskeie beheerbare en onbeheerbare veranderlikes is die belangrikste stappe hier. Ander veranderlikes soos die voorsiening van spesifieke kliëntevraag kan ook identifiseer word.

3.1.3 Konstruksie van 'n simulasiemodel

Vier van die belangrikste prosesse in die konstruksie van 'n simulasiemodel is (Chase & Aquilano; 1992: 796):

- Spesifisering van die veranderlikes en parameters;
- Spesifisering van die besluitnemingsreëls;
- Spesifisering van die waarskynlikheidsverdeling; en
- Spesifisering van tydinkremente.

Eerstens moet die navorser eienskappe identifiseer wat in die modellering kan verander (veranderlikes) en dié wat onveranderd bly (parameters). In die meeste simulasies is die fokus op die veranderlikes op verskillende tydstippe, soos die aantal ritte afgelê deur 'n sekere voertuig per maand. Nou moet die besluitnemingsreëls opgestel word. Hierdie reëls is groepe voorwaardes waaronder die gedrag van die simulasie opneembaar is.

Twee kategorieë verdelings kan vir simulasie gebruik word, *naamlik empiriese frekwensieverdelings* en *wiskundige frekwensieverdelings*. Empiriese verdelings verteenwoordig die doelgemaakte groep verdelings van byvoorbeeld aanvraag, wat slegs relevant is vir 'n spesifieke probleem. In die geval waar die aanvraag neig na 'n

bekende verdeling soos die normaal- of Poissonverdeling, sou data-insameling en insette grootliks vereenvoudigde getalle wees.

Die spesifisering van die tydinkremente kan op twee maniere geskied, naamlik met vaste of veranderlike inkrementering. In die eerste geval word uniforme tydinkremente soos minute en ure gespesifiseer, waarna die simulasie van een interval na die volgende voortgaan. Vir elke inkrement-eenheid word die model nagegaan om te kontroleer of 'n gebeurtenis gaan plaasvind. Wanneer dit wel plaasvind, word die simulasie gedoen, en indien daar niks gaan plaasvind nie, word daar bloot een inkrement-eenheid bygevoeg. By veranderlike inkremente word die modeltyd tot en met die volgende gebeurtenis aangeskuif.

Modelle met 'n groot aantal gebeurtenisse, of waar die gebeurtenisse met gereelde tussenposes plaasvind, is meer geskik vir die vaste tydinkremente. Wanneer daar 'n relatiewe klein aantal gebeurtenisse plaasvind, of wanneer gebeure ongereeld verspreid is, behoort die stelsel van veranderlike tydinkremente meer geskik te wees aangesien dit minder rekenaartyd en -geheuespasie in beslag neem. Die stelsel ignoreer tydintervalle waar daar geen gebeurtenisse plaasvind nie, en spaar sodoende bewerkingstyd en -plek in die geheue.

3.1.4 Waardes van veranderlikes en parameters

Die navorser is hier betrokke by hoofsaaklik twee aktiwiteite, naamlik die bepaling van aanvangstoestande en die bepaling van bewerkingstyd (Chase en Aquilano; 1992: 800). Aanvangstoestande speel 'n groot rol in die taktiese benadering van simulasie. Die rede hiervoor is dat die model sydig reageer as gevolg van die aanvangstoestande in 'n program, totdat dit na 'n ewewig afplat. Hierdie fase van die simulasielopie staan bekend as die oorgangs- of opwarmingstydperk (Winston; 1994: 1220). Die sydige resultate kan op verskillende maniere hanteer word, waaronder:

- Resultate in die aanvangsperiodes word geïgnoreer;
- Die seleksie van aanvangstoestande sodat die beginfase so kort as moontlik is; en
- Seleksie van aanvangstoestande wat sydigheid elimineer.

Die bepaling van die bewerkingstyd hang af van die doelwit van die simulasie en die akkuraatheid wat benodig word. Die mees algemene benadering is om die simulasie te laat voortgaan, totdat 'n ewewigstoestand bereik word. Die navorser kan ook die simulasie vir 'n gegewe tyd laat voortgaan en self beoordeel of die resultate realisties is. Dit is ook moontlik om die simulasie te laat voortgaan vir 'n lang tydperk, met die doel om statistiese hipotesetoetsing uit te voer. 'n Simulasielopie wat oor 'n relatiewe kort tydperk plaasvind, word 'n *terminerende* simulasie genoem, en 'n *ewewigsimulasie* is een wat oor 'n tydperk van sogenaamde oneindigheid 'n ewewigstoestand bereik (Winston; 1994: 1220). Die hantering en interpretasies van resultate word in afdeling 3.1.5 bespreek.

3.1.5 Interpretasies van resultate

Die akkuraatheid van die interpretasies van die modelleringsresultate word beïnvloed deur die mate waartoe die model die werklikheid voorstel. Die ontwerp van die statistiese aspek in simulasie speel ook 'n groot rol in die akkuraatheid, en sommige navorsers beskou simulasie as 'n soort hipotesetoetsing.

Die proses-analis het normaalweg ook ondersteunende data byderhand waaraan hy die akkuraatheid van sy model kan meet (Chase en Aquilano; 1992: 800). Die inligting van werklik voltooide prosesse in die stelsel, inligting van soortgelyke stelsels, en die gevoel of ervaring van die analis dien as maatstawwe vir die interpretasiefunksie. Daar is egter geen plaasvervanger vir die resultate van die werklike prosesse nie. Die enigste ware toets van 'n simulasie is hoe die werklike stelsel presteer.

3.1.6 Geldigverklaring

Die rekenaarprogram moet, voordat enige afleidings daaruit gemaak kan word, getoets word om die simulasie korrek uit te voer. Dit is dié stap waarin daar verseker word dat die vertaling van die werklike stelsel in die rekenaartaal korrek en akkuraat is. Foute in die program kom voor as gevolg van foute in die kodering of foute in die

logika. Koderingsfoute word normaalweg maklik opgespoor aangesien die rekenaar by die aanwesigheid hiervan geen bewerkings uitvoer nie.

Foute in die logika van die program is normaalweg meer kompleks, omdat die rekenaar wel die berekenings uitvoer, maar nie die regte antwoord verskaf nie. Die programmeerder kan hierdie tipe foute opspoor aan die hand van een van drie metodes, naamlik:

1. Kry drukstukke van al die bewerkings en verifieer dit per hand.
2. Simuleer die huidige toestande, en vergelyk dit met die resultate van die huidige stelsel.
3. Kies een punt in die simulasielopie, en vergelyk dit met die resultate van die wiskundige model wat daardie punt beskryf.

Waar ontwerpers die derde metode tot hul beskikking het, is simulasieresultate nie nodig nie. 'n Wiskundige model kan in meer komplekse probleme slegs enkele fasette van die probleem voorstel, en tog van nut wees vir foutopsporing.

Nuwe simulasie-eksperimente kan ook gebruik word indien die navorser dit nodig ag. Hier kan parameters, besluitnemingsreëls, veranderlikes, begintoestande, en die bewerkingstyd verander word. Alhoewel dit soms lyk asof 'n ewewigstoestand bereik is, kan 'n verlenging van die simulasietydperk aantoon dat ander resultate verkry word. Dieselfde simulasielopie oor 'n ander tydperk, kan beskou word as 'n nuwe eksperiment.

3.1.7 Parameterskatting

Vir die skatting van parameters met behulp van simulasie is daar 'n vereiste aantal simulasielopies nodig om die betroubaarheid van die parameter te verseker (Winston; 1994: 1228). Gestel dat die parameter 'n standaardafwyking van σ het, en akkuraatheid word vereis binne 'n waarde D met 'n breukdeel van 100 (1- α), dan word die aantal simulasielopies (n) benodig deur vergelyking 3.1 gegee.

$$n = z^2_{\alpha/2}\,\sigma^2 / D^2 \qquad 3.1$$

Hier bevredig z_α die verdeling van $P\ (\ Z > z_\alpha\ ) = \alpha$, en Z is normaalverdeel volgens die eienskappe $N\ (0,1)$ geldig. Hierdie waardes word uit die standaardwaarskynlikheidsverdelingstabel van die normaalverspreiding verkry. Die simbool α is die verkose betekenispeil, wat in afdeling 3.1.6 gedefinieer word.

3.2 Rekenaarmatige benaderings

Simulasie bereik die volle potensiaal daarvan wanneer dit ingespan word in die ondersteuningsraamwerk van 'n rekenaar. Tydens simulasie is daar soveel berekenings dat dit te duur sou wees om modelle sonder 'n rekenaar te bou. Die bou van 'n rekenaarmodel bestaan uit die volgende stappe:

1. Seleksie van 'n rekenaartaal;
2. Visualisering van vloeikaarte;
3. Kodering;
4. Generering van data;
5. Uitsetverslae; en
6. Geldigverklaring.

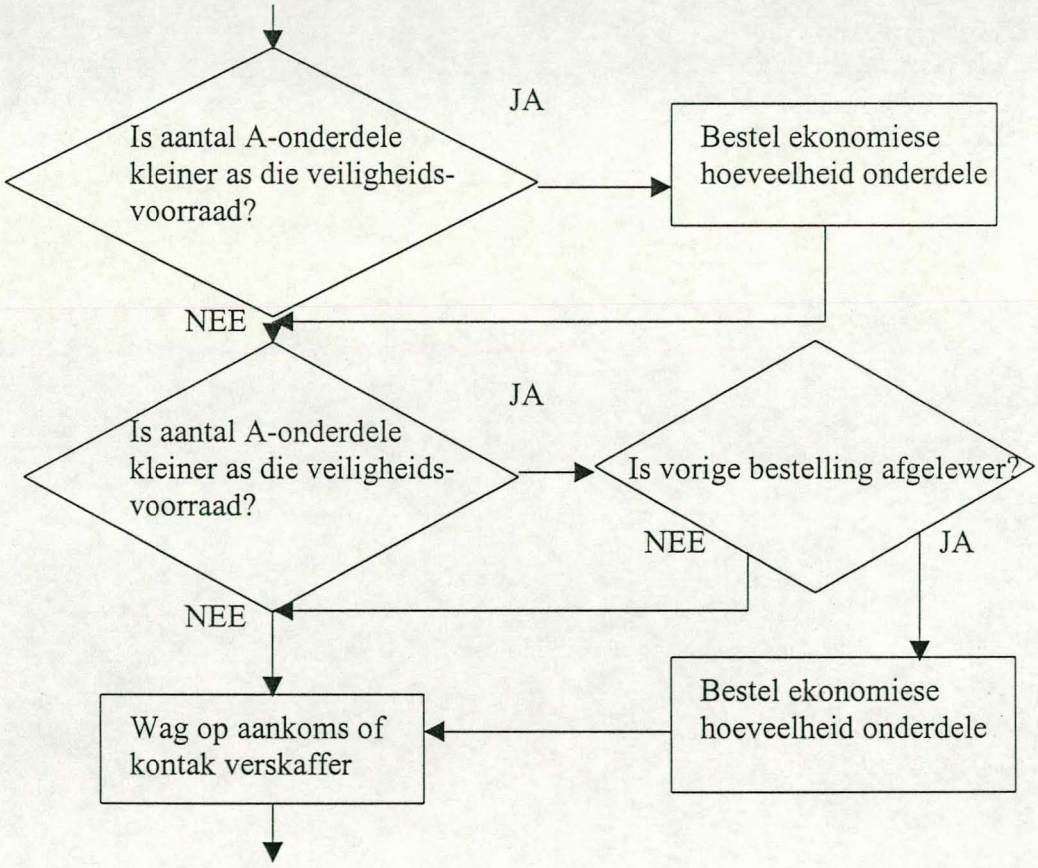
Algemene rekenaartale is hoofsaaklik FORTRAN, COBOL, Pascal, C, PL/1 en BASIC (Chase en Aquilano; 1992: 803). Hierdie tale het die voordeel dat dit in 'n wye verskeidenheid toepassings gebruik kan word. Spesialedoeltale soos FORTRAN, APL, GPSS, GASP en GERT is nie suiwer rekenaartale nie, en verskil van algemene rekenaartale. Tale soos hierdie vereenvoudig bloot die simulasieprosedure deur die gebruik van standaardsubroetines, en is meer geskik vir toue en skeduleringsprobleme as gevolg van hulle verkorte programmeringstyd.

Die visualisering van vloeikaarte behels dat die navorser die vloeipatrone onder dinamiese omstandighede visualiseer. Kodering is daardie stap waar navorsers die vloeikaart in 'n betrokke rekenaartaal vertaal. In die geval van 'n algemene rekenaartaal is die meganika wat betrokke is by die skryf van so 'n taal, in dieselfde formaat as dié van 'n wiskundige probleem. Datagenerering verwys na die normaalweg ingeboude proses van ewekansige generering van data, wat op 'n aantal doelgemaakte wyses kan geskied. Die mees korrekte metode van ewekansige

datavorming moet gekies word, met ander woorde, die metode wat die beste by die simulasiedoelwit pas.

Vloeikaarte soos die een in Figuur 3.1 is tipies van dié in die vervoeronderneming. Hier kan die reeks stappe in 'n bestellingsproses van onderdele gesien word, soos in die werkswinkel van 'n vervoeronderneming. Die bestelling van voorraad geskied slegs wanneer die voorraadvlak onder die veiligheidsvoorraad daal. Tydens die wagtydperk van voorraadbestellings moet die veiligheidsvoorraad nie meer as enigste maatstaf vir bestellings dien nie, maar moet 'n wagtydperk gebruik word. Wanneer bestellings nie afgelewer word nie, kan die onderneming langer wag of die verskaffer kontak.

Figuur 3.1 Vloeikaart van 'n bestellingsproses



Die prosedure vir die generering van prosestye vanuit die gegewe waarskynlikheidsverdelings is bekend as steekproefneming, ewekansige generering van veranderlikes, of die Monte Carlo eksperiment (Winston; 1994: 1192). Die

beginsel van diskrete verdelings is gebaseer op die interpretasies van die frekwensieverdelings en hulle waarskynlikhede. Die doel van die benadering is om langtermynuitkomste te genereer, wat voorkom volgens die frekwensies wat in die verdeling gespesifiseer is. Die steekproefverdeling moet boonop onafhanklik wees, wat beteken dat elke steekproefgetal onafhanklik moet wees van die getal wat dit voorafgaan en volg.

Uitsettipes en spesiale verslae word normaalweg in 'n algemene rekenaartaal gekies, en spesialedoeltale vereis bloot dat die programmeerder standaardroetines aktiveer om die gewenste uitsette te verkry. Te veel of te min data is algemene probleme in simulasie-uitsette, want in beide gevalle kan betekenisvolle inligting van die studie verskuil word.

Vir die simulasie van 'n hele verskeidenheid probleme is sigblaaie soos Lotus, Quattro, Excel, en SuperCalc baie handig. Groot databasisse kan hierin gereduseer word tot interpreteerbare inligting, maar sigblaaie kan ook die generering van ewekansige getalle in simulasies hanteer.

Net soos rekenaartale kan simulasietale ook geklassifiseer word volgens algemene en spesiale doelwitprogrammatuur. Voorbeelde van algemene doelwitprogramme is SLAM II, SIMSCRIPT II.5, SIMAN, GPSS/H, GPSS/PC, PCMODEL en RESQ (Chase en Aquilano; 1992: 814). Spesiale doelwitprogramme word gebruik vir meer spesifieke tipes toepassing, en voorbeelde hiervan is MAP/1 en SIMFACTORY. So sou die programmeerder vervaardigingsprosesse kan spesifiseer volgens hulle eienskappe, soos die aantal werksfunksies, aankomstempo's, prosestye, lotgroottes, hoeveelhede onvoltooide werk, en hulpbronne wat beskikbaar is. In toepassings waar verwantskappe en vloei tussen werkstasies in 'n proses ontleed moet word, is die program FACTORY baie handig. Hier word gebruik gemaak van genetiese algoritmes vir die optimale uitlegbeplanning van 'n fabriek.

3.3 Eienskappe in simulasieprogrammatuur

Aangesien die meeste simulasieprogramme 'n geruime tyd neem om baas te raak, en navorsers dus neig om nie maklik oor te skakel na 'n ander program nie, is dit raadsaam om die regte program te kies. Die navorser moet kyk na die volgende eienskappe:

1. Die vermoë om te simuleer en interaktief gebruik te word;
2. Gebruikersvriendelikheid;
3. Toelating om modelle eers te bou en later te koppel;
4. Gebruikers moet hulle eie roetines skryf en gebruik;
5. Die gebruik van ingeboude boublokkies en opdragte;
6. Makro-prosesvermoë waarvolgens *werkselle* ("machining cells") ontwikkel kan word;
7. Die vermoë om materiaal te hanteer waar veral die vervoer van materiaal belangrik is;
8. Uitsetstatistiek soos siklustye en wagtye;
9. Verskeidenheid dataontledings vir insette en uitsette;
10. Animasievermoë om die produktevloei grafies te vertoon; en
11. Aktiewe foutopsporing sodat navorsers tydens lopies die vloei kan volg en foute kan identifiseer.

Daar word in die volgende hoofstukke breedvoerig uitgewei oor die sukses van die verkose simulasieprogram aan die hand van hierdie eienskappe en ander. Dit bly egter belangrik dat navorsers hulle nie sal blindstaar teen hierdie eienskappe nie, maar eerstens sal bepaal watter eienskappe vir die betrokke simulasie benodig word.

4. Gevallestudie van die hanteringsfunksie

Die navorser het as praktiese voorbeeld vir sy projek die ontvangs-, sorterings- en versendingsaanleg van 'n koeriermaatskappy in Kaapstad bestudeer. Die sorteringsproses is gekies om in die simulasiestudie gebruik te word.

4.1 Agtergrond

Vir die modellering van enige proses moet die navorser beskik oor alle nodige inligting en eienskappe vir die programmering van die probleem. Die navorser moet deeglik bewus wees van die probleem of probleme in die aanleg, en doelwitte stel waarvolgens die simulasiestudie aangepak word. Die navorser moet waak teen die oorbeklemtoning van die doelwit, om sodoende die onsydigheid van die studie te behou.

4.1.1 Probleemdefinisie

Die onderneming benodig 'n wetenskaplike benadering tot die huidige prosesse wat bedryf word. Moderne tegnologie word reeds ingespan vir die erkenning van ontvangs en versending van items, in die vorm van 'n staafkodeleser en die kodering van alle items met staafkodes. Daar is egter 'n gebrek aan die omhulselvorming van indiwiduele werksfunksies, en 'n OG benadering behoort hierdie tekortkoming te oorbrug. Die maatskappy benodig 'n voorstelling van die vloei van goedere deur die stelsel, wat aantoon watter invloed elke werksfunksie op die vloei uitoefen.

Die hantering van items in die aanleg is slegs een van 'n reeks aktiwiteite in die voorsieningskanaal, maar dit is die kern van die onderneming. In hierdie sektor lê die oorsprong, en dus ook die oplossing van baie probleme en onnodige koste opgesluit. Die maatskappy wil die hanteringskoste per item verlaag, en beskou hierdie koste saam met die hanteringstyd as 'n maatstaf van produktiwiteit vir die aanleg.

Voorheen was daar geen maatstaf vir die prestasiebeoordeling van die werksfunksies in die aanleg nie. Met behulp van die simulasiestudie beoog die navorser om die indiwiduele prestasie van elke werksfunksie te meet.

4.1.2 Eienskappe van die aanleg

Die aanleg is hoofsaaklik 'n hanteringsaanleg in 'n stelsel van geskakelde aanlegte, wat saamwerk om die volgende take met items uit te voer:

- ontvang,
- sorteer, en
- stoor en versend.

Die hoofdoelwit van alle aktiwiteite is om 'n betroubare diens te lewer en die kliënte tevrede te stel, deur die volgende kwaliteitsaspekte aan items te verleen:

- tyd,
- toestand,
- plek, en
- waarde.

Dit impliseer dat die maatskappy moet toesien dat alle items wat ontvang word die korrekte bestemming op die regte tydstip in die korrekte toestand bereik, en dat hierdie diens gelewer word op die mees doeltreffende wyse om die laagste moontlike tariewe met inagneming van ondernemerswins toe te laat.

Die aanlegte word as eenheid bestuur volgens die *as-en-speek*-beginsel. Dit beteken dat 'n netwerk aanlegte hier elk 'n bedieningsgebied het waarheen items aanvanklik versend word. Hierna word dit gesorteer en versprei na die onderskeie bestemmings binne die bedieningsgebied, en die items wat na bestemmings binne ander gebiede versend is, word na die *as-aanleg* van daardie betrokke gebied versend. Bogenoemde versendings geskied gewoonlik in groot maat met 'n vervoereenheid wat op vasgestelde tye tussen die aanlegte items vervoer. Die items word dus outomaties onderwerp aan 'n tydsbeperking wanneer dit tussen aanlegte verskeep word. 'n *As* verteenwoordig 'n aanleg wat dien as ontvanger van items binne sy gebied, en doen

gelyktydig diens as 'n verspreider van ontvangde items vanuit sy eie gebied na die ander asse. 'n *Speek* verteenwoordig die lynvervoer van items langs vasgestelde roetes na en vanaf 'n as.

Hierdie stelsel hou voordele in vir 'n vervoerkontrakteur. Daar is baie goeie beheer oor die ontvangs- en verspreidingsproses, en 'n baie groot gebied kan hierdeur bedien word. Ligte voertuie vervoer'n kleiner hoeveelheid items op daaglikse basis oor korter afstande, terwyl swaarvoertuie oor langer afstande groter vragte tussen die aanlegte kan vervoer. So word die vervoervoordeel van elke modus gebruik, naamlik massavrag per swaarvoertuig en kortafstand, ligte vrag per ligte voertuig. Die as-aanlegte werk ook as eenheid saam, wat terugvrag van goedere vul vir die meeste ritte wat onderneem word. Dit verminder die verliese van leë vraghouers wat in die terugrigting vervoer moet word.

Die stelsel se grootste nadeel is dat dubbele vervoer kan voorkom. Aangesien aanlegte saamwerk en vrag na mekaar versend, gebeur dit dat items na en vanaf as-aanlegte oor langer paaie vervoer word as wat nodig is. Enkelkarweiers kan in hierdie mark soms laer pryse bied, omdat vervoer oor korter roetes moontlik is. Gelukkig is die hoofsentra in Suid-Afrika meestal ver van mekaar geleë, en verbindingspaaie is dikwels beperk tot enkele roetes as gevolg van fisiese struikelblokke.

'n Ander nadeel ontstaan by die foutiewe versending van items. Dit kan gebeur dat 'n item na die verkeerde as-aanleg versend word, waarna dit soms via ander roetes so spoedig moontlik na die korrekte aanleg versend word. Dit kan groot opvolgkoste en onnodige wagtye veroorsaak, wat aanlegte kan benadeel en kliënte ontevrede laat.

Die hantering van *vreemde* items, dit wil sê items onderweg na en afkomstig van 'n ander as-aanleg, kan onnodige hanteringskade en verdere foutiewe versendings tot gevolg hê, aangesien dit moeilik is om die bron van hierdie foute op te spoor. Die simulasiestudie kan nie in alle bogenoemde probleme oplossings bied nie, maar bied veral beter beheer oor entiteite, en behoort 'n laer foutiewe versendingspersentasie tot gevolg te hê.

4.1.3 Doelwitte van simulasiestudie

Die eerste doelwit van simulasie is om die werklikheid getrou na te boots, waar die ideale geval nie noodwendig beskryf word nie, maar die korrekte modeleienskappe getoon word. Met dié doelwit in gedagte kan die navorser die moontlik foute in 'n proses raaksien, en dit probeer uitlig in die simulasie sonder om die eienskappe aan te pas. Wanneer 'n navorser dus werk met die doel om bewys te lewer van sekere verwagte foute in die proses, is die resultaat gewoonlik duidelik. Simulasie is veral handig indien aanpassings aan 'n foutiewe area in die stelsel gemaak moet word, sodat die navorser voor die fisiese verandering reeds die verwagte resultate elektronies kan skat. Dit kan onnodige uitgawes aan veranderinge in die stelsel voorkom. Dit word gedoen vir die balansering van kapasiteite tussen werkstasies.

Die doelwitte van hierdie simulasie is om 'n werksfunksie vir elke personeellid te beskryf, sodat oortollige personeel of personeelfunksies met te veel ledige tyd uit die stelsel verwyder kan word. Verder is die doel om aan te toon wat die prestasie van die objekte in die aanleg is.

Die navorser het 'n standaardtydsduur vir elke werksfunksie gekies, sodat elke funksie afsonderlik evalueer kan word. Die model gebruik ook hierdie inligting om 'n realistiese nabootsing te verseker. In die meeste gevalle word die normaalverdeling gebruik vir die keuse van tydsduur.

4.2 Konstruksie van 'n model

Soos in afdeling 3.1.3 word daar eerstens gekyk na die spesifisering van veranderlikes en parameters. Daarna word die besluitnemingsreëls geformuleer, die waarskynlikheidsverdeling gekies, en laastens word die tydinkremente gespesifiseer.

4.2.1 Spesifisering van die veranderlikes en parameters

Die veranderlikes in die model is:

- Die hoeveelheid pakkies wat elke voertuig na die aanleg bring;
- Die hoeveelheid pakkies wat in elke voertuig gepak word;
- Die aantal werkstasies wat beman word; en
- Die snelheid waarmee hanteringstoerusting pakkies deur die aanleg vervoer.

Die parameters in die model is:

- Hoeveelheid voertuie wat arriveer;
- Tyd wat elke voertuig arriveer;
- Tyd wanneer houers en voertuie moet vertrek;
- Kapasiteit van elke voertuig;
- Kapasiteit van die hanteringstoerusting;
- Personeel se hanteringsnelheid van pakkies;
- Grootte en gewig van pakkies;
- Volgorde van werkstake op pakkies uitgevoer; en
- Aankomstempo van pakkies in die stelsel (normaalverdeel).

Die navorser het vir die model 'n standaardgrootte en -gewig aan elke pakkie toegeken. Pakkies het ongelyke massas en groottes, maar daar bestaan geen rou data wat hierdie eienskappe voldoende beskryf nie. Boonop verskil die hanteringstye van pakkies baie min, aangesien alle pakkies vir die werknemers relatief maklik hanteerbaar is.

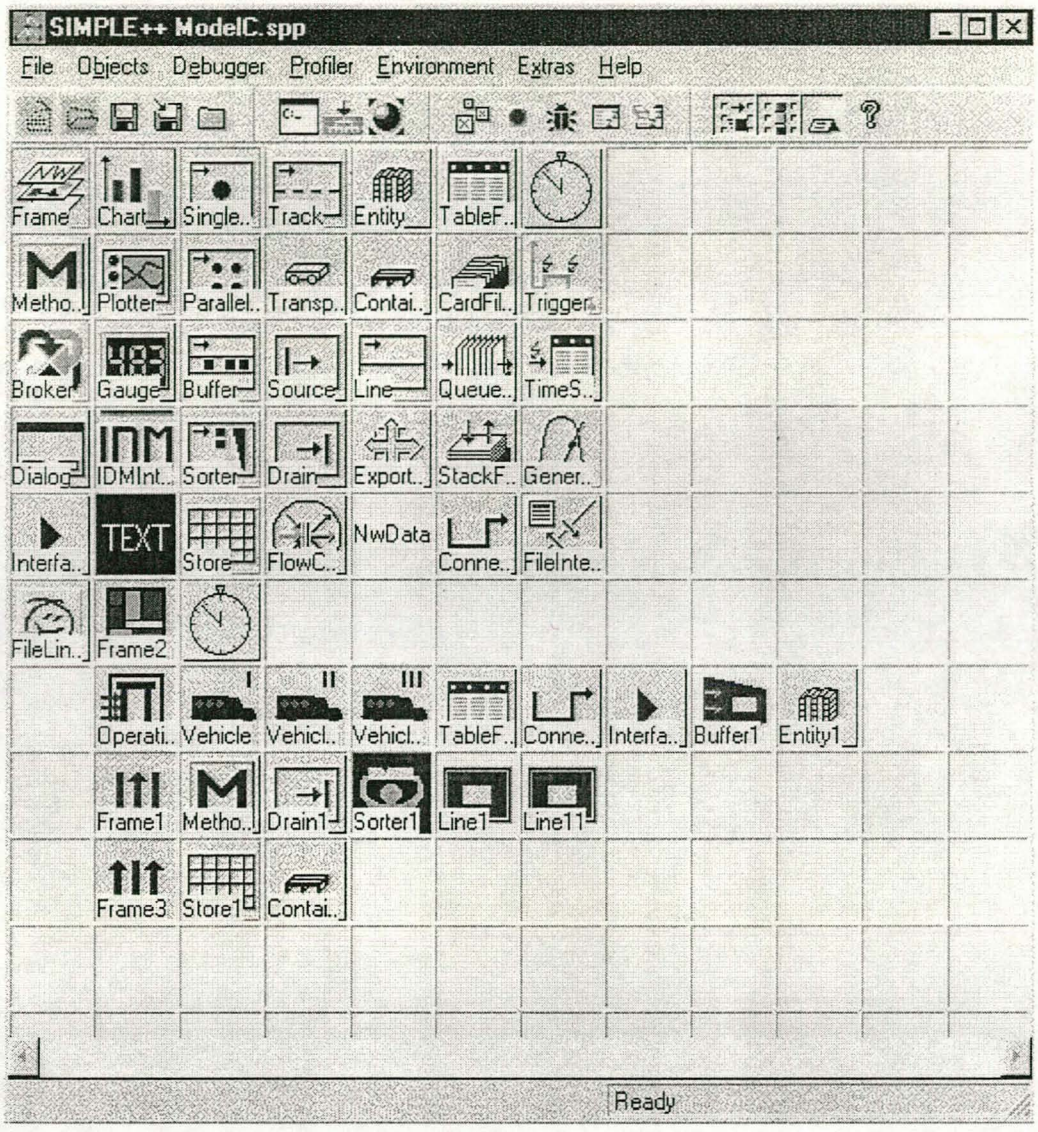
Voertuie wat arriveer en vertrek, bedien redelik konstante roetes en normaalweg dieselfde kliënte, wat hierdie tye redelik konstant maak. Dus word die aankomstye en die ideale menslike hanteringsnelheid vir pakkies gekies, sodat 'n mate van eenvormigheid in die programmeringstyd die program vereenvoudig.

Die volgende objekte word in die model gebruik, en word in Figuur 4.1 aangetoon saam met die ouerobjekte:

- sorteerder,
- drein (stoor van items na hantering),
- bron (ontvangs van items voor hantering),
- buffer (kontrolepunt van ontvangde items),
- lyn (vervoerband),
- entiteit of bewegende eenheid (item),
- raam (funksieverdeling),
- beheermeganisme (simulasielopie begin en eindig),
- tabel (lêer waarin itemfrekwensies gestoor word),
- metode (beheer oor hanteringstegnieke),
- koppellyn (verbinding tussen objekte),
- stoor (stoor van foutiewe items), en
- intervlak (koppeling tussen rame).

In Figuur 4.1 word al die moontlike objektipes in die SIMPLE++ biblioteek aangetoon, saam met die modelelemente wat die navorser in sy model gebruik het. Die boonste groep objekte is bekend as die ouerobjekte, en word deur die program voorsien. Die onderste groep objekte is geskep vanuit die ouerobjekte, waarvan sommige voorsien word van doelgemaakte simbole. Die nuutgevormde objekte behou egter die eienskappe van die ouerobjekte, behalwe in die gevalle waar die navorser kies om sekere eienskappe te verander.

Figuur 4.1 Objekbiblioteek en modelobjekte in SIMPLE++



Die items wat in die stelsel hanteer moet word, staan bekend as entiteite of bewegende eenhede. Dit beskik oor 'n standaardgrootte, naamlik 'n lengte, breedte, en hoogte van 0.75 meter. Hierdie fisiese grootte is gekies as 'n maatstaf om die kritieke geval te beskryf wat betrekking het op die kapasiteit van 'n vervoerband. Vervoerband A wat in Figuur 4.2.5 aangetoon word, is een van die bottelnekke in die stelsel. Daar bestaan 'n kapasiteitsprobleem teen die snelheid wat die band beweeg, wat deur die grootte van die entiteite beïnvloed word. 'n Kritieke gemiddelde breedte van 0.75 m verteenwoordig die geval waar daar benutting van die band plaasvind, deur elke 0.75 meter 'n entiteit te plaas. Die vorm van entiteite word vereenvoudig deur aan te neem dat elke entiteit 'n vierkantige eenheid is, met die bogenoemde afmetings. Hierdie

vereenvoudiging is aanvaarbaar, aangesien die fisiese grootte van entiteite net op die vervoerbande in ag geneem word.

Die lyne wat die vervoerbande voorstel, is elk 9 m lank en beweeg teen snelhede van onderskeidelik 0.3 en 0.7 m/s. Die vervoerbandstelsel word dus voorgestel as langwerpige eenhede wat slegs die op- en aflaai van entiteite aan die eindpunte daarvan toelaat. Die werklike situasie stem baie hiermee ooreen, en min eienskappe gaan deur hierdie aanname verlore. Twee lyne naamlik *Line1* en *Line2*, maak dit moontlik vir die programmeerder om deur middel van oorerflikheid aanpassings aan die snelhede te maak, net soos dit in die aanleg gedoen word.

Elke sorteringsobjek verteenwoordig een aflaaipunt in die stelsel, wat deur ander werknemers bedryf word. So byvoorbeeld word die aflaaipunt van entiteite na die lughawe deur drie personeellede bedryf wat die aflaaiwerk, kontrole, en pakwerkfunksies behartig. Indiwiduele eienskappe kan aan elke objek toegeken word om die beste moontlike beskrywing te bied, maar as gevolg van die beperking in simulasietyd word slegs een standaardwerkstyd aan elke objek toegeken. Dit sou ideaal wees indien daar vir elke werknemer 'n objek ontwikkel kan word, maar as gevolg van onderstaande twee redes word dit nie gedoen nie.

1. Die studentelisensie van die pakket SIMPLE++ beperk die gebruiker tot 'n sekere aantal objekte, wat oorskry word indien daar vir elke werknemer een objek ontwikkel word.
2. Die doelwit van OG programmering is nie om noodwendig vir elke geval 'n unieke program te ontwikkel nie, maar om 'n standaardmodel te bou wat met behulp van 'n klein aantal aanpassings geskik is om 'n verskeidenheid modelle te beskryf. Hierdeur word programmeringstyd geminimeer.

Die buffers in die stelsel beskryf die kontrolepunte van die ontvangs en verspreiding van entiteite, wat tans deur werknemers met elektroniese staafkodelesers beman word. Elke entiteit word by plasing op die vervoerband ingelees, en weer ingelees wanneer dit die stelsel verlaat as dit in 'n houer of op 'n palet verpak word. Sodoende word 'n databasis bedryf waar alle in- en uitvloei van entiteite gekontroleer word. Die hoofdoelwit van hierdie stelsel is nie om kontrole uit te oefen oor die volledigheid van

besendings nie, maar om diefstal te voorkom, en om die oorsake van groot logistieke probleme te ondersoek.

Die rame wat in die model gebruik word, verdeel die ontvangs-, sorterings- en verspreidingsfunksies. Elke funksie word in een raam geplaas, waarvandaan toegang daartoe weer verkry kan word. Die onderskeie rame is verbind met behulp van intervlakke, en elke raam kan vier intervlakke hê. Sodoende kan die linker-, regter-, bo-, en onderkante van elke raam met 'n ander verbind word om die vloei van entiteite of inligting moontlik te maak. Hierdie tegniek word gebruik om die model eenvoudiger en visueel verstaanbaar te maak. Figuur 4.3 toon duidelik die verbindings tussen rame aan met twee verbindings tussen ontvangs en sorterings, en een verbinding tussen sorterings en die stoor van entiteite.

Die beheermeganisme word in die hoofraam geplaas, en beheer die simulasielopies deur dit te begin en te eindig. In die beheermeganisme is dit moontlik om die tydsduur en die opneem van resultate te reguleer, sodat die betrokke situasie getrou gevolg kan word. Dit is ook moontlik om simulasies te doen van stelsels wat in lande met verskillende tydsones bedryf word.

Bronne neem 'n spesifieke tipe entiteit herhaaldelik vanuit 'n tabel in die model. Die tabel bevat die name en frekwensies waarin entiteite voorkom, en stel dit beskikbaar aan dié objekte waarin die tabel as 'n ontvangste-objek gekies is. Seleksie vanuit die tabel vind op ewekansige wyse plaas.

Metodes word gebruik en tussen objekte verdeel, maar beskryf in die meeste gevalle die manier waarop 'n aktiwiteit uitgevoer word. Die C++ taal word in die metodes gebruik, en spel die logika van die model uit. 'n Metode in SIMPLE++ is soortgelyk aan 'n reeksopdrag (*makro*) in sigbladprogramme. Beide funksies voer 'n opeenvolgende ketting opdragte uit, wat varieer na gelang van die nakoming, al dan nie, van sekere geprogrammeerde vereistes.

Die laaste vesting is 'n stoor wat bloot ingesluit word indien die model defektiewe entiteite genereer. Dit is hoofsaaklik van nut in die programmeringsfase van die model, of kan gebruik word wanneer aanpassings aan die model gemaak moet word.

'n Stoor - en nie 'n drein nie - word gebruik omdat die programmeerder enige ongewensde entiteit wat daarin beland, wil uitken, en sodoende die oorsaak van die probleem kan opspoor.

4.2.2 Spesifisering van die besluitnemingsreëls

Daar bestaan 'n hele aantal voorwaardes wat bevredig moet word voor die simulasie betekenisvolle resultate sal lewer. Data wat gebruik word vir die aantal items wat daagliks ontvang word, is verkry vanuit die werklike ontvangsdata vir 'n tydperk van drie maande. Hierdie tydperk word beskou as 'n bron van redelike verteenwoordigende data, aangesien dit in die tyd van die jaar gekies is waarin die onderneming gemiddeld vertoon het. Die navorser vertrou die ervaring en opinie van die operasionele bestuurder in hierdie opsig.

Die ontvangs- en verspreidingsfunksie van die aanleg word vir die daaglikse tydperk 17:00 tot 22:00 gesimuleer. Hierdie proses sluit die hele hanteringsiklus van die maatskappy in, aangesien die omgekeerde proses hiervan tussen 02:00 en 07:00 daagliks plaasvind. Die simulasie kan dus geldig wees vir beide prosesse indien die programmeerder 'n aantal klein veranderinge aan die program aanbring. Waar die eerste nagskof hoofsaaklik te doen het met die ontvangs en hantering van items, het die tweede nagskof hoofsaaklik met die hantering en versending van items te doen. Vir die doel van hierdie werkstuk word slegs die aanvanklike ontvangs en hantering van entiteite in die genoemde interval beskou.

Die seisoenale tendens vir die vraag na vervoerdienste in die bedryf word geïgnoreer, en die model word vir die gemiddelde geval opgestel. Met addisionele aanvraagsyfers sou dit moontlik wees om die model aan te pas vir seisoenspieke, maar dit is nie die studiedoelwit nie. Daar word slegs 'n model benodig vir die gemiddelde aanvraagsituasie, waarmee die maatskappy strategiese besluite kan neem. Piektoestande kan hanteer word deur die kontraktering van tydelike werkers en 'n verhoogde vervoerbandsnelheid.

Entiteite word slegs in enkelformaat tot die proses toegelaat. Entiteite kan mekaar nie verbysteek vandat dit die vervoerband bereik nie, en word weer in enkelformaat by die onderskeie stasies uit die stelsel verwyder. Die simulasiemodel strek oor die tydperk vanaf die oomblik dat 'n entiteit vanuit die voertuig gelaai, en dus ontvang word, totdat dit weer in die versendingsvoertuig of op die palet geplaas word.

Daar word aanvaar dat elke werknemer oor dieselfde fisiese vermoë beskik en entiteite binne dieselfde tydperk hanteer. Entiteitgrootte en -gewig is normaalweg sodandig dat een persoon dit in dieselfde tydperk as enige ander entiteit kan hanteer. Vir die enkele uitsonderings maak die mislukkingsfunksie in elke werkstasie voorsiening. Elke entiteit beslaan dieselfde grootte area, alhoewel die meeste items heelwat kleiner is as die gespesifiseerde grootte.

Hierdie area is bloot die spasie wat elke item op die vervoerbande benut, en is 'n goeie maatstaf vir die aanvaarbare en hanteerbare verpakkingsdigtheid van items op die vervoerband. 'n Digter verpakking van items sal kongestie op die hanteringspunte en foutiewe versending van items veroorsaak. Werkstasies kan nie teen die gemiddelde vervoerbandsnelheid meer eenhede hanteer as wat die eenheidsgrootte op die bandspasie toelaat nie.

Die onderskeie funksies van inligtinghantering word saam met die hanteringstyd vir elke item in berekening gebring. Die oorhoofse hantering van inligting word nie in hierdie simulasielopie in berekening gebring nie, aangesien dit onafhanklik van die sorteringsproses kan geskied.

4.2.3 Spesifisering van die waarskynlikheidsverdeling

Volgens Keller en Warrack (1997:294) is die normaalverdeling 'n getroue voorstelling van die steekproefgemiddelde wat op ewekansige wyse uit enige populasie geneem word, mits die steekproef relatief groot is. Dit staan ook bekend as die sentrale limietstelling, en die akkuraatheid daarvan berus op die waarskynlikheidsverdeling van die ouerpopulasie, en van die grootte van die steekproef.

Die normaalverdeling word dus gebruik om die aankomstempo van items in die model te beskryf. Die waardes vir die gemiddelde en standaardafwyking is gekies op grond van die toetsresultate van die aankoms en die mening van die werknemers. Die model word oor 'n tydperk van vier tot vyf uur gesimuleer, wat verseker dat die steekproef redelik groot is, en ooreenstem met die tydperk van daaglikse aktiwiteite in die aanleg.

In Tabel 4.1 word 'n volledige lys van al die verkose verdelings gegee. Die verdelings word saam met spesifieke modelname vir elke objek in die model toegeken. Waar daar twee of meer objektipes van dieselfde klas gebruik word, is dit bloot om onderskeid te tref tussen sekere eienskappe van die klas. So byvoorbeeld verskil die begintydstip van die bronne, aangesien voertuie wat die stelsel betree, nie almal gelyktydig opdaag nie, maar oor 'n tydperk van 'n uur by die aanleg items aflewer. Waar tye volgens die normaalverdeling beskryf word, word dit gegee as N ($\bar{x}$, σ, BG, OB) vir die skatting van die parameter μ. BG en OG is die boonste en onderste grense vir die vertrouenspeil. 'n Betekenispeil van $\alpha = 0.05$ word gebruik, wat impliseer dat die proporsie van waardes met herhaalde steekproefneming van die gegewe populasie tussen die boonste en die onderste intervalgrense gelyk sal wees aan 0.95 of 95%. In die eksperiment verseker hierdie α-waarde dat betekenisvolle resultate verkry sal word. Tye word in die formaat van ure, minute, sekondes, en breukdele van sekondes gegee.

Tabel 4.1 Opsomming van verkose waarskynlikhede

Objek-klas	Objek-tipe	Proses-tyd	Opstel-tyd	Herstel-tyd	Siklus-tyd	Begin tyd	Kapa-siteit
Drein	**Drain**	N(5,2,4.2,5.8)	0	0	-	0	1
Bron	**VehicleA**	N(15,5,13,17)	0	-	-	0	1
	VehicleB	N(15,5,13,17)	0	-	-	30:00	1
	VehicleC	N(15,5,13,17)	0	-	-	1:30:00	1
Buffer	**Buffer1**	Konstant: 2.00	0	0	N(3,1,2.6,3.4)	-	1
	Buffer11	Konstant: 2.00	0	0	N(3,1,2.6,3.4)	-	1
Sorteeder	**Sorter**	-	0	N(3,1,2.6,3.4)	N(5,1.5,4.4,5.6)	-	3
Lyn	**Line1**	30	0	0	-	-	12
	Line2	12.8571	0	0	-	-	12

4.2.4 Spesifisering van tydinkremente

Normaalweg kan dit aanvaar word dat die model ewewig bereik na 'n simulasielopie gelykstaande aan een jaar se ontvangs. Die maatskappy werk 50 weke per jaar teen vyf dae per week. Ewewig word dus bereik na 250 skofte van vyf werksure per skof, wat in die program as 52 volle dae en twee uur hanteer word. Die navorser kies die aanvangstoestand om so spoedig moontlik programewewig te bereik.

Toetslopies het egter getoon dat die resultate vir een gesimuleerde skof van vyf uur resultate lewer wat getrouer is aan die werklikheid. Die ongereguleerde aankoms van voertuie en sekere vloeiprobleme met die stelsel veroorsaak 'n ongebalanseerde en oneweredige vloei van entiteite in die stelsel. Op die tydstip waar programewewig in die simulasielopie bereik word, is daar 'n eweredige vloei van produkte wat nie die werklikheid voorstel nie. Die doelwit van 'n akkurate nabootsing word wel bereik in die skoftydperk van vyf uur, wat dus gebruik word vir die simulasietydperk.

'n Vaste basis vir die beoordeling van resultate word gekies, waarmee die navorsing verskeie afleidings kan maak. Die simulasie word in tydinkremente verdeel, naamlik die tydperk van een sortering. Hierdie tydperk word gebruik om die model te beoordeel, aangesien dit verteenwoordigende resultate lewer vir die sorteringsaktiwiteite van een dag.

4.2.5 Fisiese uitleg van die stelsel

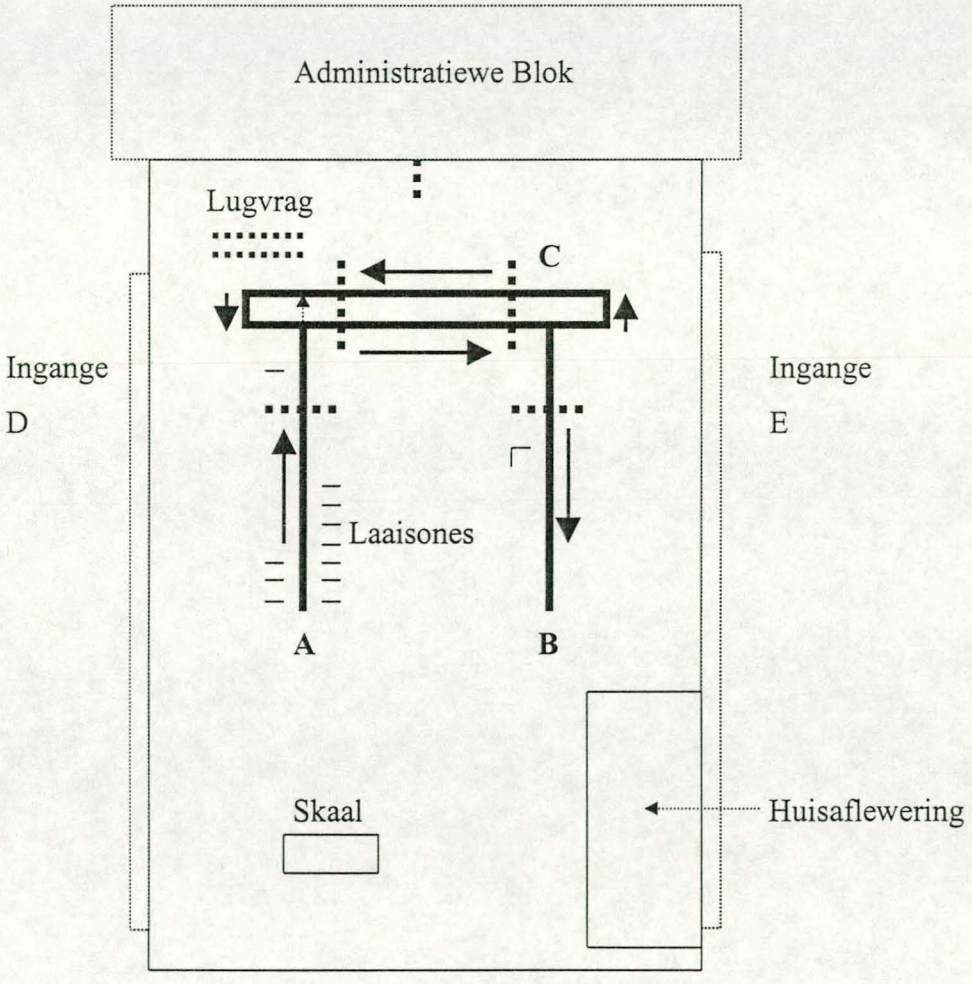
Die maatskappy beskik oor sy eie aanleg wat verdeel is in die administratiewe en uitvoerende werksfunksies. Die administratiewe ondernemingsfunksie word nie in die studie ondersoek nie, maar wel die hanteringsfunksies op die fabrieksvloer en die verwante vloei van inligting. Figuur 4.2 toon 'n vloerplan van die gesimuleerde area. Die vervoerbande, die onderskeie deponerings- en versendingsareas, sowel as die werknemersfunksies is die hoofkomponente van die studie.

Goedere word per voertuig deur ingange D en E in Figuur 4.2 vervoer, afgelaai by die onderskeie laaisones, en op die vervoerbande geplaas. Hierna volg goedere 'n vaste

roete deur die aanleg, en word by die korrekte versendingspunte van die vervoerbande verwyder. Dit word dan in voertuie of op palette gestoor. Sommige goedere word eers geweeg voor dit by die laaisones in die hanteringstelsel gevoeg word.

Die maatskappy bedryf ook sy eie huisafleweringsdiens, bekend as HDS. Die entiteite wat bestem is om deur hierdie diens hanteer te word, kry 'n spesiale etiket en word eers in 'n afdeling vir huisaflewerings gestoor. Die entiteite word ook met die vervoerband na die korrekte afdeling vervoer, waar dit ontvang en per hand verder hanteer word.

Figuur 4.2 Rowwe vloerplan



Sleutel:

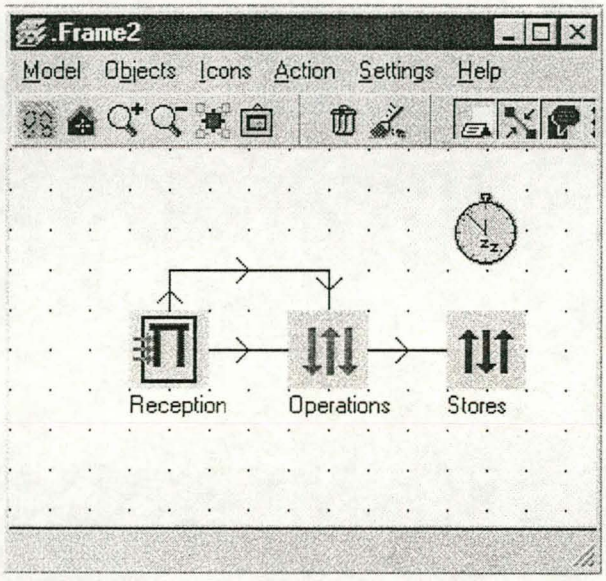
Beweegrigting van goedere op vervoerbande =

Trappe =

4.2.6 Gesimuleerde uitleg

Die fisiese uitleg word nageboots deur 'n ineenskakeling van drie subraamobjekte binne een raam. In Figuur 4.3 word die hoofraam getoon, waarvandaan toegang tot die subrame verkry kan word. Die hoofraam beskik ook oor die beheermeganisme vir die simulasie, wat deur die simbool van 'n horlosie voorgestel word. Met hierdie objek word die simulasielopie en opname van die resultate beheer, en die simulasielopie kan op enige oomblik gestop of finaal beëindig word.

Figuur 4.3 Hoofraam van die stelsel



5. Resultate en aanbevelings

Navorsers kan simulasieresultate gebruik en hanteer soos werklike uitsetresultate, maar nie as optimale resultate van 'n stelsel beskou nie. Die interpretasie van resultate geld ook net vir die betrokke toestande waaronder die simulasie-eksperiment uitgevoer word. Wanneer die programmeerder enige eienskap van 'n objek binne die stelsel verander, behoort die hele simulasielopie herhaal te word vir die interpretasie van resultate.

5.1 Hoofkomponente

Die resultate van die simulasie bestaan hoofsaaklik uit die volgende datakomponente vanuit die program:

- Aantal items hanteer deur elke werkstasie en die kapasiteit daarvan;
- Totale aantal items deur die stelsel hanteer;
- Ledige tyd en werkstyd van die werkstasies;
- Relatiewe besetting, wagtyd, en werktyd; en
- Kapasiteit van elke werkstasie.

Simulasie is nie 'n optimeringsmeganisme nie, maar dit is die plig van die navorser om 'n simulasieformaat te ontwikkel waarin resultate verkry word wat die naaste aan optimaal kan kom. Die fisiese struktuurveranderinge kan op hierdie sub-optimale stelsel gebaseer word. Tydens simulasie van die hanteringstelsel van die maatskappy, het die navorser daarin geslaag om 'n sub-optimale stelsel te ontwikkel.

5.2 Simulasieresultate en gevolgtrekkings

In Tabel 5.1 en 5.2 word 'n opsomming van die simulasieresultate oor 'n tydperk van vyf uur verskaf. Die volgende aspekte van die resultate word bespreek, wat ook as kolomme in die tabelle opgeneem is:

- ontvangs,
- versendings,

- persentasie onbenutte tyd,
- relatiewe besetting,
- persentasie wagtyd,
- persentasie werktyd,
- persentasie geblokkeerde tyd,
- persentasie mislukkings, en
- kapasiteitsbenutting.

5.2.1 Ontvangste

Die aantal keer dat elke objek ontvang word, weerspieël die frekwensies waarin entiteite in die stelsel moet voorkom, en hierdie frekwensies is ingesluit in die tabel waaruit alle bronne hulle entiteite neem. Wanneer die programmeerder die frekwensies verander, is die resultaat duidelik te bespeur in die verandering van die aantal keer dat entiteite voorkom. Met die simulasie van 'n vier- tot vyfuur-periode, word nagenoeg die korrekte aantal entiteite vir elke adres gegenereer. Die ontvangsfunksie in die model is ontwikkel met die doel om 'n realistiese uitset aan die hanteringsfunksies te gee, en nie noodwendig om die korrekte fisiese uitleg na te boots nie. Die verwagte resultaat is bereik van entiteite wat op ewekansige wyse geselekteer en na die onderskeie adresse versend word.

Die totale aantal gesimuleerde keer van ontvangs van 4447 en 13 entiteite stem ooreen met die werklike aantal keer van ontvangs per dag. Hierdie ontvangs is verkry uit die data van *Buffer 1* en *Buffer 2* in Tabel 5.1, en verteenwoordig die somtotaal van die entiteite wat uit die bronne afkomstig is.

5.2.2 Versendings

Die aantal versendings vind plaas in ooreenstemming met die aantal keer van ontvangs en die inhoud van elke objek. Uit hierdie kolom in Tabel 5.1 is dit duidelik dat die vervoerbande wat deur lyne voorgestel word, verreweg die grootste aantal entiteite in hulle stelsel behou, en ook die stadigste deel van die hanteringsisteem is.

Ander aktiwiteite soos sortering en verpakking van entiteite neem baie minder tyd in beslag.

Daar word in die simulasielopie oor vyf uur 4443 entiteite versend, wat ooreenstem met die werklike aantal versendings per dag. Hierdie syfer is verkry uit die totale aantal versendings van die sorteringsobjekte, en is een maatstaf waarvolgens die doeltreffendheid van die modellering gemeet kan word. In die geval waar die gesimuleerde aantal versendings nie ooreenstem met die werklike aantal versendings nie, is die model nie 'n realistiese nabootsing van die werklikheid nie. Hierdie syfer hoef egter nie presies gelyk aan 'n werklike gemiddelde versendingsyfer te wees nie, aangesien entiteite op ewekansige wyse geselekteer en hanteer word.

5.2.3 Persentasie onbenutte tyd

Objekte in die ontvangsfunksie van die model is gemiddeld 30.08% van die tyd onbenut. Die syfer is realisties omdat daar by verskeie geleenthede tydens die proses oponthoude is, hoofsaaklik vanweë twee redes:

1. Die wisselvallige aantal keer van aankoms van entiteite, en
2. Die frekwensie en voorkoms van wagtye wat veroorsaak word deur vervoerbande wat vir kort tye afgeskakel is.

Die onbenutte syfer uit Tabel 5.2 van 97.77% van objekte in die sorteringsfunksie van die model is onafwendbaar, aangesien sorteerders moet wag op entiteite om hulle werkstasies te bereik. Dit gebeur dikwels dat 'n werkstasie deurentyd beman moet word vir die enkele geleentheid wanneer 'n entiteit op daardie punt uit die stelsel geneem word. Die huidige vervoerbandstelsel laat min ruimte vir die balansering van kapasiteit. Die ideale proses sou meer ruimte laat vir een sentrale distribusiepunt, waar die adres van entiteite slegs een keer nagegaan word, en dit op dieselfde punt verdeel word na die onderskeie eindpunte. Na afloop van die verdeling van entiteite na hulle eindpunte, kan die verpakkingsfunksie deur een groep werknemers voltooi word. So kan die onderneming ten volle gebruik maak van sy werkersmag, en dubbele werk voorkom deur slegs een adresleesfunksie te bedryf.

5.2.4 Relatiewe besetting

'n Gemiddelde relatiewe besettingsyfer van 48.48% vir die ontvangsfunksie is aanvaarbaar. Een probleem in die stelsel word egter duidelik wanneer die relatiewe besetting van die bronne nagegaan word. *VehicleA1* word baie meer gebruik as *VehicleD1*, hoofsaaklik vanweë die versadiging van die stelsel. Op die eerste deel van die vervoerband bestaan daar volop spasie vir die plasing van entiteite, maar die bronne wat laer af in die vloeikringloop hulle entiteite op die band moet plaas, ondervind probleme om spasie op die band te kry. Uit die resultate blyk duidelik dat die bronne se relatiewe besetting daal vanaf die beginpunt van die vervoerband tot by die eindpunt. Hierdie probleem kom ook in die werklikheid voor.

Die versadigingsvlak van die band kan beïnvloed word deur die snelheid waarteen dit loop. Die snelheid is verstelbaar, maar band A moet stadiger as band B en C gestel word om 'n bottelnek te voorkom (Figuur 4.2). Hoe vinniger die band loop, hoe hoër is die kapasiteit van die stelsel, en hoe meer entiteite kan vervoer word. Teen 'n hoër snelheid sal daar dus minder wagtyd by ontvangs en meer spasie op die band wees.

Die relatiewe besettingsgraad van die bande vanaf *Line1* tot *Line17* daal drasties van 26.8% tot 1.77%. Soos entiteite van die band verwyder word, raak dit al leër en eindig op 'n 0% besetting van *Line18*. *LineA* in die ontvangsfunksie het 'n besettingsgraad van 99.55%, wat net deur 'n mislukking verhoed word om 'n 100% benutting te hê.

Die lae besettingsgraad van *Buffer2* is nie van belang nie, aangesien die werksfunksie van hierdie objek aangevul word deur die hantering van alle entiteite wat die koppelvlak tussen band A en C van Figuur 4.2 oorkruis. Die besettingsgraad van *Buffer2*, die elektroniese staafkode-kontrolefunksie, is meer realisties met 'n syfer van 49.41%.

Oor die algemeen het die sorteerders 'n onrealistiese lae besettingsgraad, wat hoofsaaklik veroorsaak word deur die keuse van 'n ideale werkstempo. Dit bevestig egter ook dat die sorteerders baie laag beset word, aangesien die aard van die werk sodanig is dat werknemers verplig word om by een werkstasie te wag op entiteite om

hulle te bereik. Die 0% besettingsgraad van *Line19*, *Sorter42*, *Sorter44* en *Store11* dui net daarop dat geen entiteite dié betrokke objekte bereik nie, en die model korrek funksioneer.

5.2.5 Persentasie wagtyd

Wagtye speel nie 'n baie groot rol in die evaluering van die model nie, aangesien objekte soos sorteerders as deel van hulle taak op entiteite wag. Wagtyd is wel 'n maatstaf by die vervoerband se objekte. Die stygende neiging in die persentasie wagtyd van *Line1* tot *Line18* dui op een van die wanbalanse in die stelsel. Die vervoerbande is sodanig ontwerp dat namate die proses vorder, dit al hoe leër raak. Dit is dus onvermydelik dat die bande aan die einde van die vloeikringloop feitlik leeg is, en langer wagtye hier veroorsaak word.

5.2.6 Persentasie werktyd

Die persentasie wat die hoeveelheid beskikbare tyd aandui van 'n objek wat gewei word aan werksaktiwiteite is van geen waarde by objekte wat nie werkstake verrig nie, maar dui wel op die nut van sommige objekte. So kan byvoorbeeld onderskei word tussen die nut van *Drain1* en *Drain2*, en of dit moontlik sal wees om slegs een objek te gebruik om hierdie funksies voor te stel. Teen die huidige werkstempo sal dit nie moontlik wees nie, aangesien die som van die werktydpersentasies 100% oorskry.

5.2.7 Persentasie geblokkeerde tyd en mislukkingstyd

Hierdie funksie maak voorsiening vir toestande in die produksie waar 'n objek met sy aktiwiteite nie kan voortgaan nie, omdat die daaropvolgende objek nog nie sy take voltooi het of spasie het vir 'n entiteit nie. Dit is 'n baie algemene verskynsel in die praktyk, en kom voor veral wanneer 'n groot aantal funksies in serie bedryf word. Die vervoerband veroorsaak hierdie probleem, wat duidelik te bespeur is in die hoë waardes vir blokkasies in *LineA*, *LineB*, *LineC* en *LineD*. Soos te verwagte is daar 'n

baie lae voorkoms van blokkerings in die sorteringsfunksie, en slegs *Sorter32* word daardeur geraak.

Die mislukkingsfunksie van ontvangs word geaktiveer om onvoorsiene omstandighede voor te stel wat die proses stop. Die persentasie mislukkings van die ontvangstefunksie is redelik laag, met 'n hoogste waarde van 0.54% vir *Line17*. Daar kan aanvaar word dat 'n sekere persentasie falings in feitlik enige stelsel kan voorkom, en die programmeerder moet die invloed daarvan op die stelsel in ag neem.

5.2.8 Kapasiteitsbenutting

Die toelaatbare kapasiteit vir elke objek is geprogrammeer, sodat die maksimum aantal entiteite wat 'n objek dra later hiermee vergelyk kan word om te bepaal tot watter mate die kapasiteit daarvan benut word. Die doel hiervan is om die kapasiteit te balanseer, sodat alle objekte optimaal benut word. Om die kapasiteit van 'n werkstasie te verhoog kos geld, maar deur die voorsiening van 'n laer beskikbare kapasiteit kan geld bespaar word. Die maatskappy wil verhoed dat sekere werkstasies hulle kapasiteit ten volle benut, terwyl ander weer nie die beskikbare kapasiteit benut nie. Die kapasiteit moet dus tussen die onderskeie werkstasies balanseer word.

LineC en *LineD* benut nie hulle beskikbare tyd nie, terwyl die voedingslyne soos *LineA* en *LineB* wel hulle volle kapasiteit benut. Die onderneming behoort die kapasiteit van die voedingslyne te verhoog, of 'n kleiner uitgawe aan te gaan deur 'n laer kapasiteit in *LineC* en *LineD* te voorsien. *Line16* en *Line17* beskik ook oor spaarkapasiteit, wat daarop dui dat die maatskappy onnodige uitgawes hier aangaan of reeds spandeer het.

Tabel 5.1 Simulasieresultate van ontvangsfunksie

Objek-naam	Versend	Ont-vang	Leeg (%)	Rel. besetting	Wagtyd (%)	Werktyd (%)	Geblok-keer (%)	Maks. Aantal	Misluk-king (%)
Drain1	2040	2041	42.62	57.38	46.62	57.38	0	1	0
Drain2	2402	2402	33.58	66.42	33.58	66.42	0	1	0
Buffer1	4447	4447	50.59	49.41	50.59	49.41	0	1	0
Buffer2	13	13	99.86	0.14	99.86	0.14	0	1	0
LineA	312	324	0	99.55	0	8.82	90.78	12	0.4
LineB	1850	1862	0	97.75	0	45.14	54.53	12	0.33
LineC	3393	3405	0	99.92	0	47.29	52.27	13	0.44
LineD	4460	4472	0	97.26	0	62.2	37.39	13	0.41
VehicleA1	60	61	0	100	0	0	100	1	0
VehicleA2	30	31	10	90	10	0	90	1	0
VehicleB1	223	224	20	80	20	0	80	1	0
VehicleB2	315	315	0.25	99.75	0.25	0	99.75	1	0
VehicleC1	480	480	44.24	55.76	44.24	0	55.76	1	0
VehicleC2	352	352	54.32	45.68	54.32	0	45.68	1	0
VehicleD1	359	359	84.34	15.16	84.34	0	15.66	1	0
VehicleD2	238	238	86.41	13.59	86.41	0	13.59	1	0
Line1	4443	4447	0.31	26.8	0.31	99.41	0	5	0.28
Line11	4285	4288	0.38	26.1	0.38	99.3	0	6	0.32
Line12	3779	3782	0.84	23.41	0.88	98.31	0	6	0.51
Line13	3463	3465	2.07	21.09	2.2	97.63	0	6	0.17
Line14	3086	3087	3.29	19.17	3.63	95.93	0	6	0.44
Line15	2563	2566	6.46	17.85	9.17	90.48	0	6	0.35
Line16	1147	1148	33.63	8.05	40.54	59.1	0	5	0.37
Line17	270	270	79.19	1.77	81.45	18	0	3	0.54
Line18	0	0	99.68	0	99.68	0	0	0	0.32
TOTAAL	**44010**	**44079**	**752.06**	**1212.01**	**768.45**	**994.96**	**735.41**	**105**	**4.88**
GEM	**1760.4**	**1763.1**	**30.0824**	**48.4804**	**30.738**	**39.7984**	**29.4164**	**4.2**	**0.1952**

Tabel 5.2 Simulasieresultate van sorteringsfunksie

Objek naam	Versend	Ontvang	Leeg (%)	Rel. besetting	Wagtyd (%)	Werktyd (%)	Geblokkeer (%)	Maks. Aantal	Misluking (%)
Sorter1	87	87	99.44	0.52	100	0	0	1	0
Sorter11	27	27	99.5	0.17	100	0	0	1	0
Sorter12	41	41	99.16	0.28	100	0	0	1	0
Sorter13	13	13	99.8	0.07	100	0	0	1	0
Sorter14	57	57	99.1	0.3	100	0	0	1	0
Sorter15	163	163	96.83	1.1	100	0	0	2	0
Sorter16	172	172	96.18	1.32	100	0	0	2	0
Sorter17	111	111	97.81	0.74	100	0	0	2	0
Sorter18	88	88	98.6	0.47	100	0	0	1	0
Sorter19	88	88	98.27	0.59	100	0	0	2	0
Sorter20	25	25	99.51	0.16	100	0	0	1	0
Sorter21	113	113	97.7	1.00	100	0	0	2	0
Sorter22	55	55	98.98	0.34	100	0	0	2	0
Sorter23	78	78	98.63	0.46	100	0	0	1	0
Sorter24	152	152	97.21	0.93	100	0	0	2	0
Sorter25	91	91	97.88	0.71	100	0	0	1	0
Sorter26	7	7	99.84	0.05	100	0	0	1	0
Sorter27	75	75	98.51	0.51	100	0	0	2	0
Sorter28	43	43	99.71	0.21	100	0	0	2	0
Sorter29	395	395	91.68	2.85	100	0	0	2	0
Sorter30	238	238	95.82	1.41	100	0	0	2	0
Sorter31	409	409	93.74	2.17	100	0	0	2	0
Sorter32	437	437	90.89	3.26	99.94	0	0.06	3	0
Sorter33	331	331	93.67	2.2	100	0	0	2	0
Sorter34	132	132	98.1	0.65	100	0	0	2	0
Sorter35	286	286	98.28	1.61	100	0	0	2	0
Sorter36	324	324	93.72	2.15	100	0	0	2	0
Sorter37	135	135	97.25	0.98	100	0	0	2	0
Sorter38	244	244	95.12	1.68	100	0	0	2	0
Sorter39	7	7	99.86	0.05	100	0	0	1	0
Sorter40	12	12	99.77	0.08	100	0	0	1	0
Sorter41	7	7	99.97	0.01	100	0	0	1	0
Line19	0	0	99.46	0	100	0	0	0	0
Sorter42	0	0	100	0	100	0	0	0	0
Sorter43	0	0	100	0	100	0	0	0	0
Store11	0	0	100	0	100	0	0	0	0
TOTAAL	**4443**	**4443**	**3519.99**	**29.03**	**3599.94**	**0**	**0.06**	**52**	**0**
GEM	**123.41**	**123.41**	**97.77**	**0.8063**	**99.998**	**0**	**0.00166**	**1.4444**	**0**

5.3 Aanbevelings vir beter kontrole

Die maatskappy kan 'n standaardwerksprosedure en -werksmetode ontwikkel en gebruik waarin sover moontlik standaardtye vir die spesifieke take vereis word. Die aard van die werk maak dit moontlik vir ledige werknemers om deurgaans besig te wees met aktiwiteite wat nie waarde tot die produk toevoeg nie. Wanneer elke werknemer binne 'n stel reëls sy werk moet verrig, behoort die stelsel 'n hoër kapasiteit en werkstempo te hê.

5.4 Algemene strukturele aanpassings

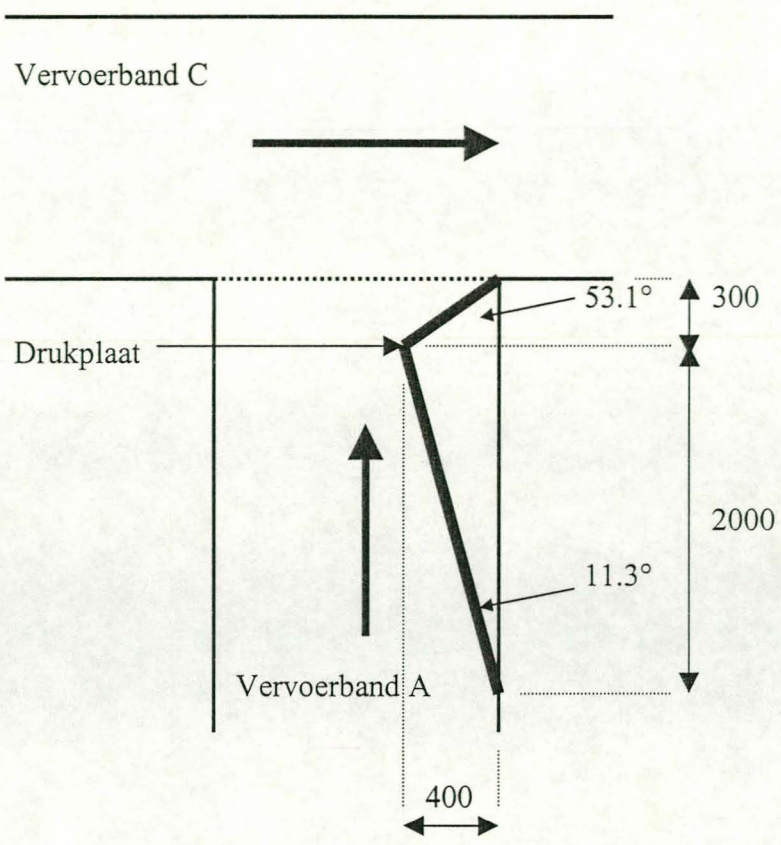
Die navorser het in die loop van die studie 'n paar stelselprobleme identifiseer, na aanleiding waarop sekere aanbevelings gemaak kan word. Hierdie aanbevelings is spesifiek gerig op die uitleg van die hanteringstelsel.

5.4.1 Vervoerbandkoppeling

Vervoerbande A van Figuur 4.2 voed vervoerband C, en op hierdie koppeling ontstaan daar verskeie probleme. Wanneer 'n aantal groter items kort op mekaar volg, ontstaan daar 'n opeenhoping by die koppeling, aangesien die reghoekige rigtingverandering in die vloei nie glad plaasvind nie. In Figuur 5.1 word die gebruik van 'n drukplaat aangetoon, wat items meer glad oor die koppeling sal laat vloei. Tans word 'n reghoekige plaat gebruik wat voorkom dat items van die vervoerband afgestoot word, en op die vloer beland.

Die nadeel van so 'n drukplaat is dat die kapasiteit van twee parallelle eenhede nou verlaag word na slegs een, as gevolg van die beperking op die wydte van die vervoerband se eindpunt. Hierdie nadeel moet egter opgeweeg word teen die alternatiewe gebruik van 'n addisionele werknemer, wat items gedurig met die hand moet regskuif om die gladde vloei daarvan te verseker. Die bestaande reghoekige plaat kan bloot gebuig en gemonteer word teen die aangetoonde hoek om die vloei te verbeter.

Figuur 5.1 Voorgestelde drukplaat vir beter vloei



Vervoerband C word teen 'n hoër snelheid as vervoerband A bedryf om 'n bottelnek in die koppelvlak te voorkom. Deur die gebruik van 'n drukplaat kan die kapasiteit van die bottelnek verhoog word, wat beter resultate behoort te lewer wanneer die onderskeie vervoerbandsnelhede gelyk aan mekaar is.

5.4.2 Elektroniese vervoerbandbeheer

Die maatskappy gebruik 'n vervoerbandbeheerstelsel wat die noodstop en aanskakeling van elke vervoerband afsonderlik vereis. In die geval van 'n vloeiprobleem sal 'n werknemer tipies vanaf die posisie wat hy beman die vervoerband stop. In die geval van 'n vloeiprobleem op die vervoerband wat die ander hanteringsfunksies voed, sal 'n noodstop bloot die toevoer van nuwe items afsny. Waar daar verder in die proses probleme met die hanteringsfunksies ondervind word, en 'n noodstop uitgevoer word, gaan die toevoer van items vanuit die eerste vervoerband voort.

Opeenhopings van items vind dus plaas in die koppelvlak tussen opeenvolgende vervoerbande. Dit kan vermy word deur 'n beheerstelsel wat verseker dat een noodstop alle meganiese vloei van items verhoed deur alle vervoerbande te stop. Dit kan ook vanaf 'n sentrale beheerpunt geskied, maar moet verkieslik vanaf die vervoerband self beheer word. Dit verseker dat probleme, waarvan die omvang sodanig is dat dit die vloei van entiteite vertraag, by die oorsprong daarvan opgelos word. 'n Kenmerk van gehalte in produksie is die oplos van probleme by die oorsprong daarvan. Gehalte word dus vanuit die kern van produksie verseker, en word meer doelmatig toegepas as deur byvoorbeeld inspeksies na afloop van produksielopies. Sodoende word die probleme by die oorsprong daarvan behandel, wat normaalweg die kwaliteit van 'n diens verhoog.

Daar moet ook toegesien word dat die bande teen 'n optimale snelheid loop, en nie noodwendig teen 'n gemaklike snelheid nie. 'n Belangrike beperking op die kapasiteit van die stelsel is die snelheid van vervoerband A van Figuur 4.2, wat weer die snelhede van vervoerbande B en C beïnvloed.

5.5 Nuwe Ontwerp

Daar is verskeie redes vir die herontwerp van die bestaande stelsel, wat hieronder bespreek word. Dit is egter nie die doel van hierdie navorsing om 'n volledige studie van 'n verbeterde hanteringstelsel te doen nie, maar bloot om 'n ander ontwerp vir die verspreidingsaanleg voor te stel.

5.5.1 Probleme met die bestaande stelsel

Die huidige uitleg van die aanleg skep vloei- en kapasiteitsprobleme, wat die maatskappy geld kos. Tans word entiteite hoofsaaklik op een punt in die stelsel geplaas, en op verskeie punte weer daaruit weggeneem. By elkeen van hierdie punte word die adres op die entiteit nagegaan om te verseker dat elke entiteit by sy bestemming uitkom. Dit gebeur dus dat die adres op een entiteit tot 32 keer nagegaan kan word (32 sorteerders) voor dit die korrekte aflaaipunt bereik. Hantering van entiteite behoort tot die minimum beperk te word.

Werknemers by die aflaaipunte moet soms lank wag voordat hulle entiteite uit die stelsel verwyder en dit verder kan hanteer. Hierdie ledige tyd kos die maatskappy geld, maar voeg geen waarde by die produk nie. Vir hierdie studie is die produk 'n entiteit wat op die regte tyd en in die regte toestand die korrekte aflaaipunt bereik.

Sommige entiteite neem etlike minute om uiteindelik die betrokke aflaaipunt te bereik. Die entiteit is dus lank in die stelsel, en neem gedurende prosestyd een kapasiteitseenheid in beslag. Hoe langer die entiteit neem om die stelsel te verlaat, hoe meer beskikbare kapasiteit word daardeur in beslag geneem.

5.5.2 Uitleg

Deur die hele stelsel te herontwerp word die bogenoemde probleme hanteer, maar die koste-implikasies van die kapitaaluitleg word nie in ag geneem nie. Figuur 5.2 stel 'n aanleg voor waar die adresse by een ontvangspunt slegs een keer nagegaan word. Al die werknemers sal aanvanklik by hierdie werksfunksie hulle take verrig, en namate die sorteringaktiwiteit vorder, kan hulle een vir een beweeg na die verpakking- en sorteringpunte waarna die entiteite versend word.

Een ontvangspunt verseker beter kontrole, aangesien entiteite slegs op hierdie punt die stelsel kan binnedring, en weer van dieselfde punt versprei word. Die gemiddelde tydsduur van 'n entiteit in die stelsel, totale hanteringstyd, die aantal adreskontrolefunksies, en werknemers se ledige tyd kan verminder word. Hierdie aannames kan in 'n verdere studie bewys of weerlê word.

Die stelsel kan so gebou word dat die ontvangspunt op 'n hoër vlak geleë is as die verpakking- en sorteringspunte. Glybane kan gebruik word om entiteite na afloop van sortering te vervoer. Wanneer die sorteringsfunksie voltooi is en die glybane gevul is met entiteite, kan dieselfde personeel wat entiteite volgens die adresse sorteer het, nou ook die verpakking behartig. 'n Ruwe vloerplan van die nuwe ontwerp word in Figuur 5.2 verskaf.

Die vervoerbandnetwerk is die grootste verskil tussen die voorgestelde en die bestaande stelsel. Die patroon van een lang netwerk word deur verskeie kort vervoerlyne vervang wat gevoed word deur een sentrale vervoerband. Die kapasiteit van hierdie stelsel sal hoofsaaklik bepaal word deur die sorteringstempo van die voedingsaar en die verpakkingstempo van die vervoerlyn met die grootste aantal entiteite.

5.5.3 Probleme met 'n nuwe stelsel

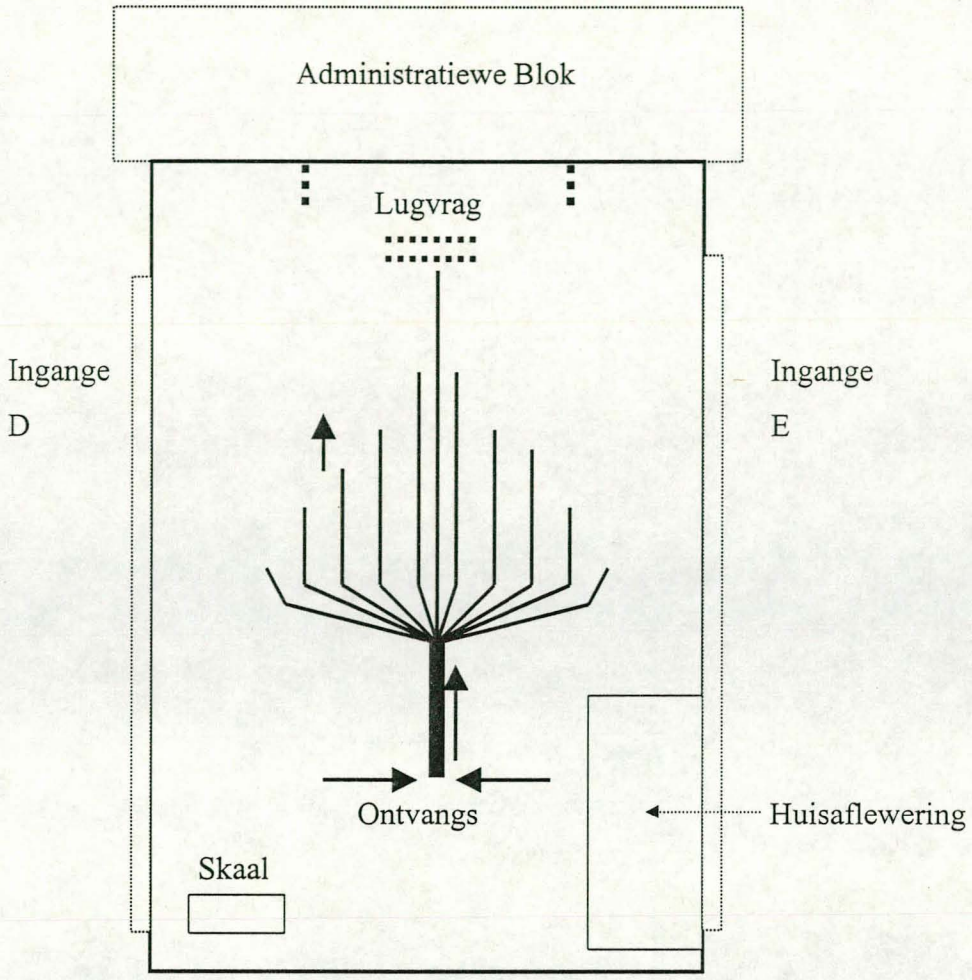
Waar werknemers in die verlede ledige tyd gehad het, kan dit nou gebeur dat daar 'n hoër totale voertuigwagtyd by die enkele ontvangspunt ontstaan. Dit kan wel beperk word deur goeie ontvangstegnieke en voldoende stoorplek by die ontvangspunt. Boonop is dit slegs een bestuurder van 'n voertuig wat wag, en nie etlike hanteringspersoneellede nie.

Die personeel kan opgelei word om 'n meer buigsame werksfunksie te verrig, om nou die ontvangs, sortering, en verpakking van entiteite te behartig. Sodoende kan personeellede na bottelnekke wat in die stelsel ontstaan, verskuif word, en die werksfunksies daar ook verrig. Opleiding kos egter geld en vergroot die beleggingswaarde van die veranderinge. Boonop word beter opgeleide personeel soms beter vergoed, wat die uitgawe aan lone kan verhoog.

5.5.4 Navorsingsgeleentheid

Dié model bied 'n geleentheid aan 'n persoon om verdere navorsing vir die ontwikkeling van 'n nuwe model te doen. Hierdie studie bewys nie die geldigheid van die voorgestelde ontwerp nie. Die verdere ontwerp en geldigverklaring daarvan kan in 'n volgende studie onderneem word.

Figuur 5.2 Rowwe vloerplan van nuwe ontwerp



Sleutel:

Beweegrigting van goedere op vervoerbande =

Trappe =

Verwysings

AESOP GmbH. 1998. *SIMPLE++ 5.0 Getting Started Manual.* Eerste Uitgawe. Stuttgart: Fraunhofer IPA.

AESOP GmbH. 1998. *SIMPLE++ 5.0 Reference Manual.* Eerste Uitgawe. Stuttgart: Fraunhofer IPA.

AESOP GMbH. 1997. *SIMPLE++ Version 4.1 Tutorial.* Stuttgart: Fraunhofer IPA.

Caspers, J. 1994. *Object Oriented Programming: Analysis, Design and Implementation Methods*. New York: Computer Technology Research Corp.

Chase, R.B. en Aquilano, N.J. 1992. *Production & Operations Management*. Sesde Uitgawe. New York: Irwin Publishers.

Council of Logistics Managament. 1996. *What's it all about.* Illinois.

Cronje, J.N. 1996. *Transport and Logistics: The Opportunity for Improved Efficiencies*. Pretoria: Departement van Vervoerekonomie en logistiek, UNISA.

Dittrich, K.R., Dayal, U. & Buchmann A.P. (Reds.) 1991. *On Object-Oriented Database Systems*. Stuttgart: Springer-Verlag.

Gattorna, J. 1990. *The Gower Handbook of Logistics and Distribution Management.* Vierde Uitgawe. Vermont: Gower Publishing Company Ltd.

Islam, N. 1994. *Distributed Objects*. IEEE Inc. New York: Computer Society Press.

Jacobson, I., Ericsson, M. & Jacobson, A. 1995. *The Object Advantage.* Stockholm: Addison-Wesley Publishing Company.

Keller, G. & Warrack, B. 1997. *Statistics for Management and Economics*. Vierde Uitgawe. Johannesburg: Duxbury Press.

Kim W. & Lochovsky, F.H. 1989. *Object-Orientated Concepts, Databases and Application.* Stuttgart: Addison-Wesley Publishing Company.

Lambert, D.M. & Stock, J.R. 1993. *Strategic Logistic Management*. Derde Uitgawe. Homewood, ILL: Irwin.

Salvendy, G. (Red) 1992. *Handbook of Industrial Engineering*. Institute of Industrial Engineers. New York: John Wiley & Sons Inc.

Verster, T. 1998. SIMPLE++ Opleidingskursus. *Opleidingskursus vir Programhantering en Simulasie in SIMPLE++*. Stellenbosch.

Vollman, T.E., Berry, W.L. & Whyback, D.C. 1992. *Manufacturing, Planning and Control Systems*. Derde Uitgawe. New York: Richard D. Irwin Inc.

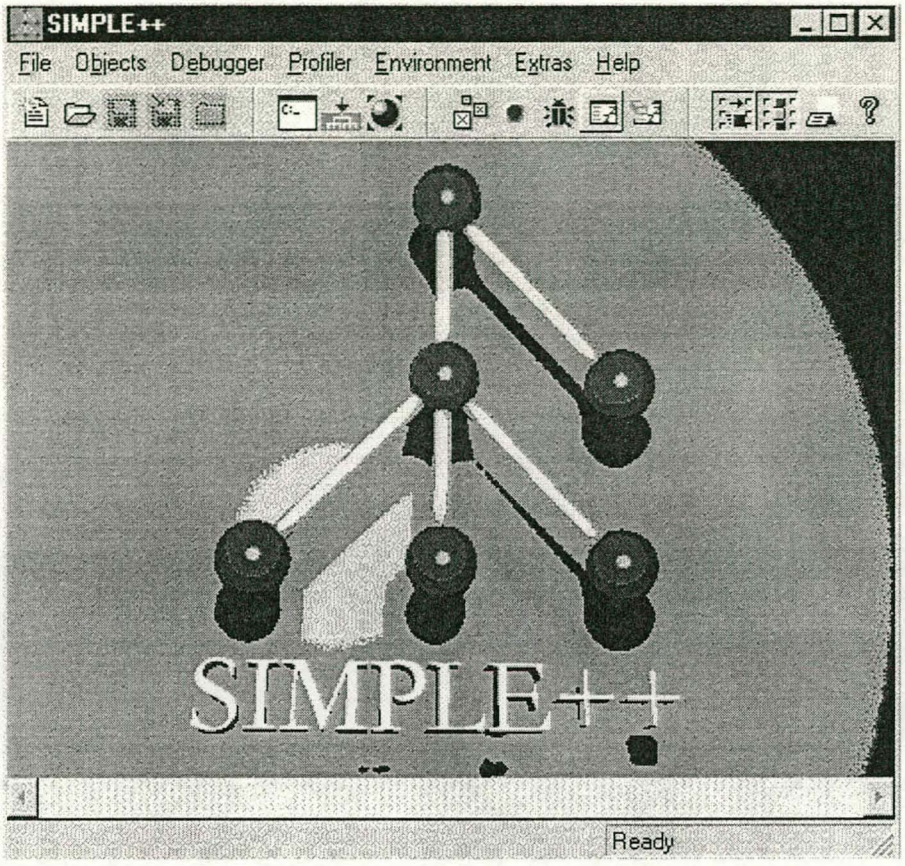
Winston, W.L. 1992. *Operations Research: Applications and Algorithms*. New York: Duxbury Press.

Zeggelink, E. 1993. *Strangers into Friends*: *The Evolution of Friendship*. Amsterdam: Thesis Publishers III.

BYLAE A

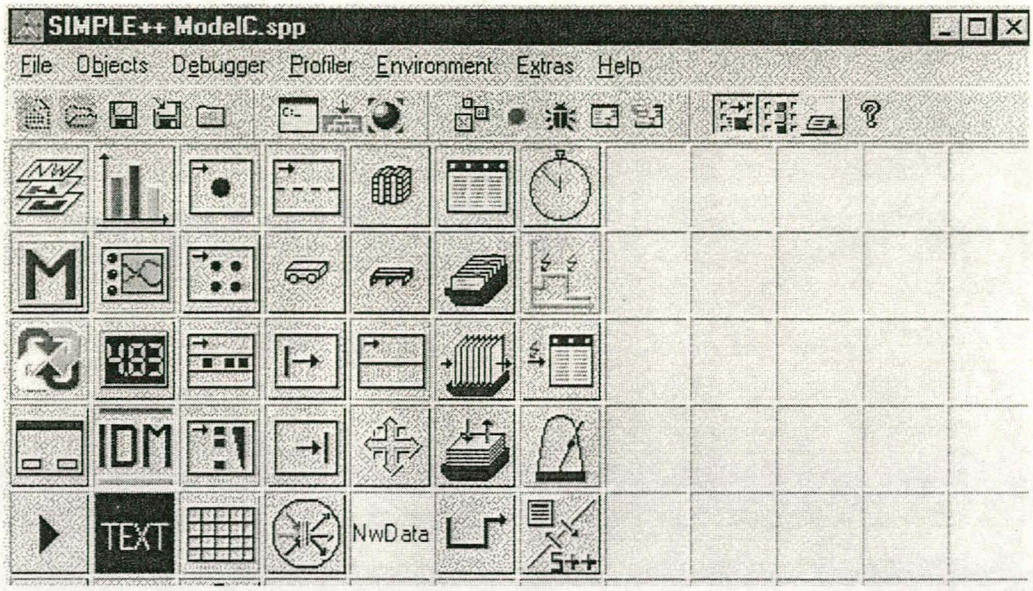
In hierdie afdeling is ten minste een voorbeeld per objekklas van elke data-invoeraksie en elke data-afvoervenster ingesluit. Onder elke venster of groep vensters word 'n kort beskrywing gegee van die werking, funksie, en interaksie van die getoonde data. Die rede vir die insluiting van hierdie programvensters is nie om die leser vertroud te maak met die programmering van sodanige stelsel nie, maar slegs om die resultate te bevestig, en om die gedagtegang te verduidelik.

Figuur 1 Hoofprogramvenster van die program



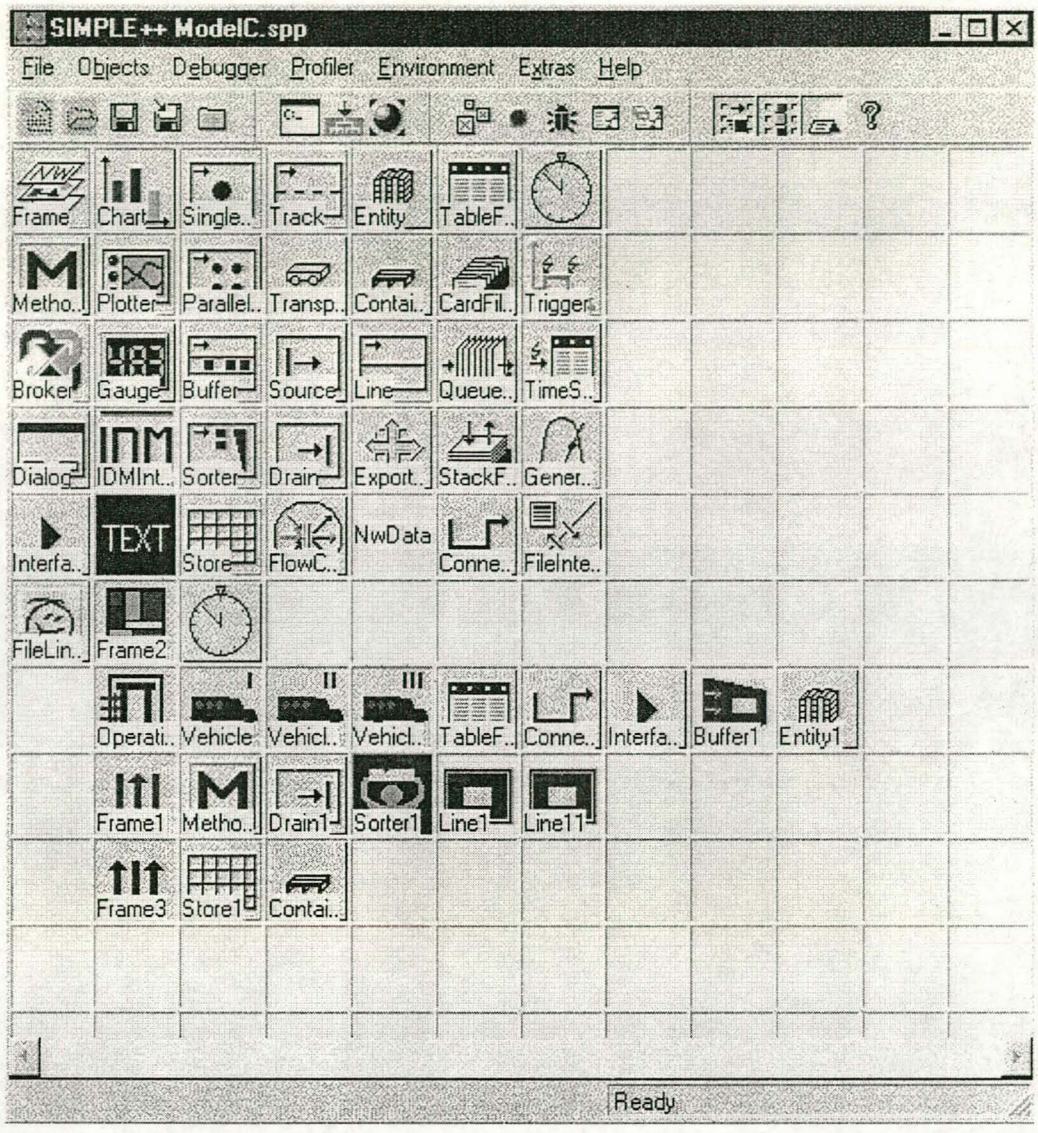
Die hoofprogramvenster word vanuit lêers oopgemaak en sekere omgewingsvoorkeure word aangetoon. Hierdie venster is die eerste venster wat verskyn wanneer die gebruiker die program aktiveer.

Figuur 2 Programbiblioteek van objekte



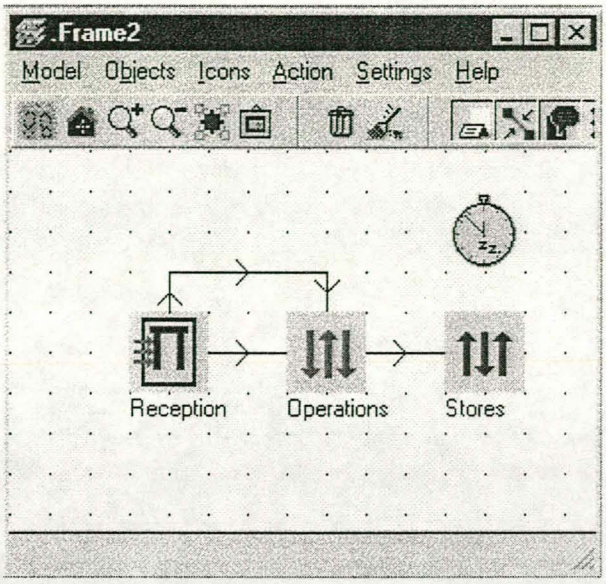
Wanneer die gebruiker 'n nuwe lêer oopmaak, toon die program hierdie venster waaruit alle moontlike objektipes gekies kan word.

Figuur 3 Modelbiblioteek vir die programmeerder



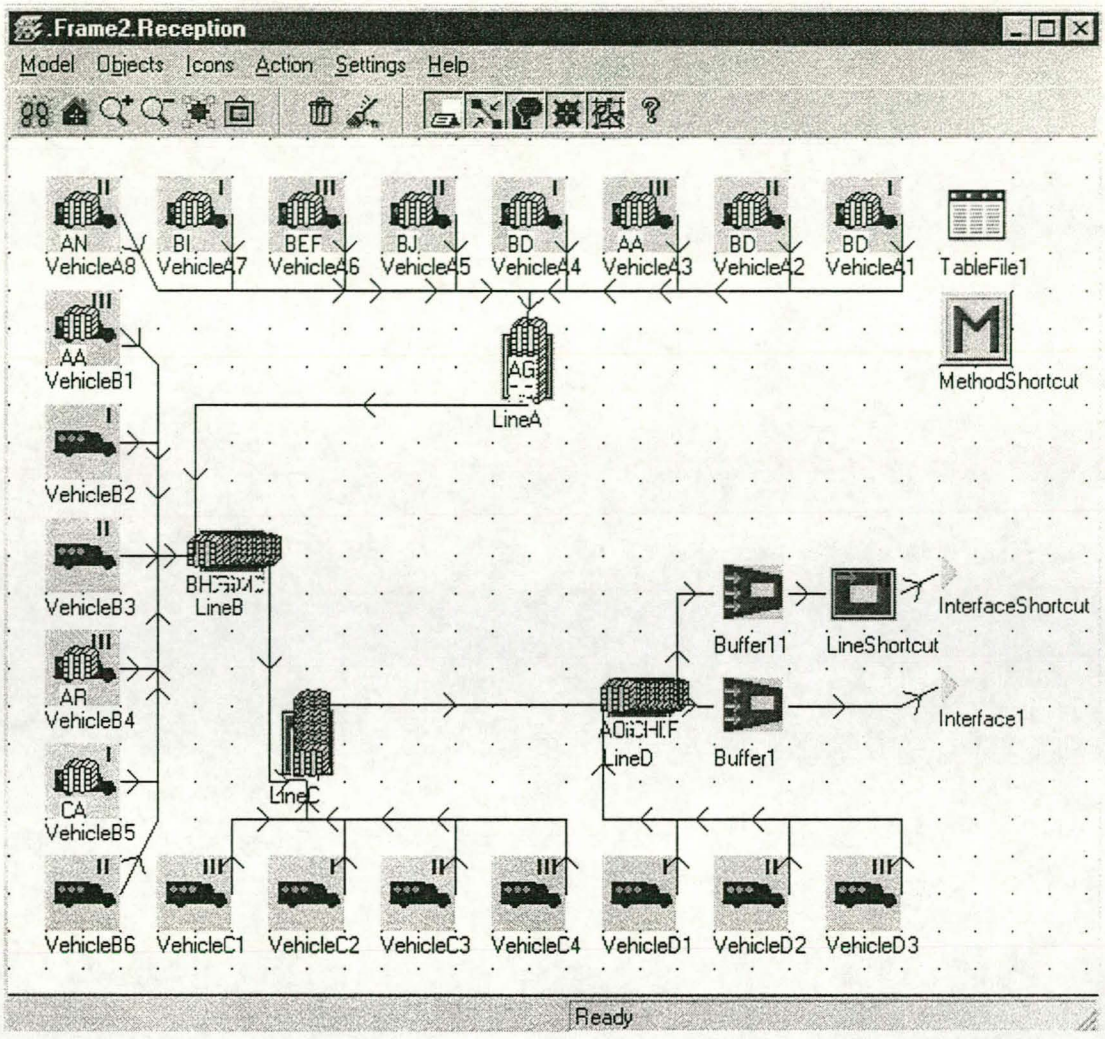
In Figuur 3 word die modelbiblioteek van *ModelC* getoon, die model wat deur die navorser ontwikkel is. Die afgeleide objekte is sigbaar in die onderste gedeelte van die venster. Elke objek hier is afgelei van 'n objek in die boonste gedeelte van die venster, alhoewel die simbole (ikone) in sommige gevalle verskil. So byvoorbeeld is die bronne die objekte genaamd *VehicleA*, *VehicleB* en *VehicleC*, wat elk vanuit die objek genaamd *Source* afgelei is. Die navorser het hier drie klasse objekte gekies aangesien die aankomstye van voertuie in drie kategorieë verdeel word.

Figuur 4 Hoofraam van die stelsel



Die venster van *Frame2* word ook in die werkstuk ingesluit, maar word vir die verduideliking hier weer aangetoon. *Frame2* is die hoofraam van die model, waarin die rame vir ontvangs, sortering, en store geplaas en verbind word. Die beheermeganisme word in die hoofraam geplaas om te verseker dat alle objekte binne hierdie stelsel gelyktydig geaktiveer en gedeaktiveer word.

Figuur 5 Ontvangs in die raam Reception



Die raam vir ontvangs beskik oor die versameling van alle bronne en die voorstelling van die eerste gedeelte van die vervoerband. Die doel van hierdie raam is om 'n stroom entiteite aan die sorteringsraam op die mees getroue wyse te voorsien. Entiteite word op ewekansige wyse vanuit die tabel geselekteer, en volgens die korrekte skedule en teen die regte frekwensie, oorgedra na die stelsel op die korrekte gedeeltes van die vervoerband. *LineA*, *LineB*, *LineC*, en *LineD* stel vier sektore op die vervoerband voor waar entiteite die stelsel kan binnedring. Dit is vir die navorser moontlik om op enige tydstip in die simulasie presies te bepaal op watter punt van die band 'n entiteit is, sodat die vordering van entiteite deur die stelsel akkuraat gemeet kan word.

Figuur 6 'n Onbenutte ontvangspunt in die raam Reception

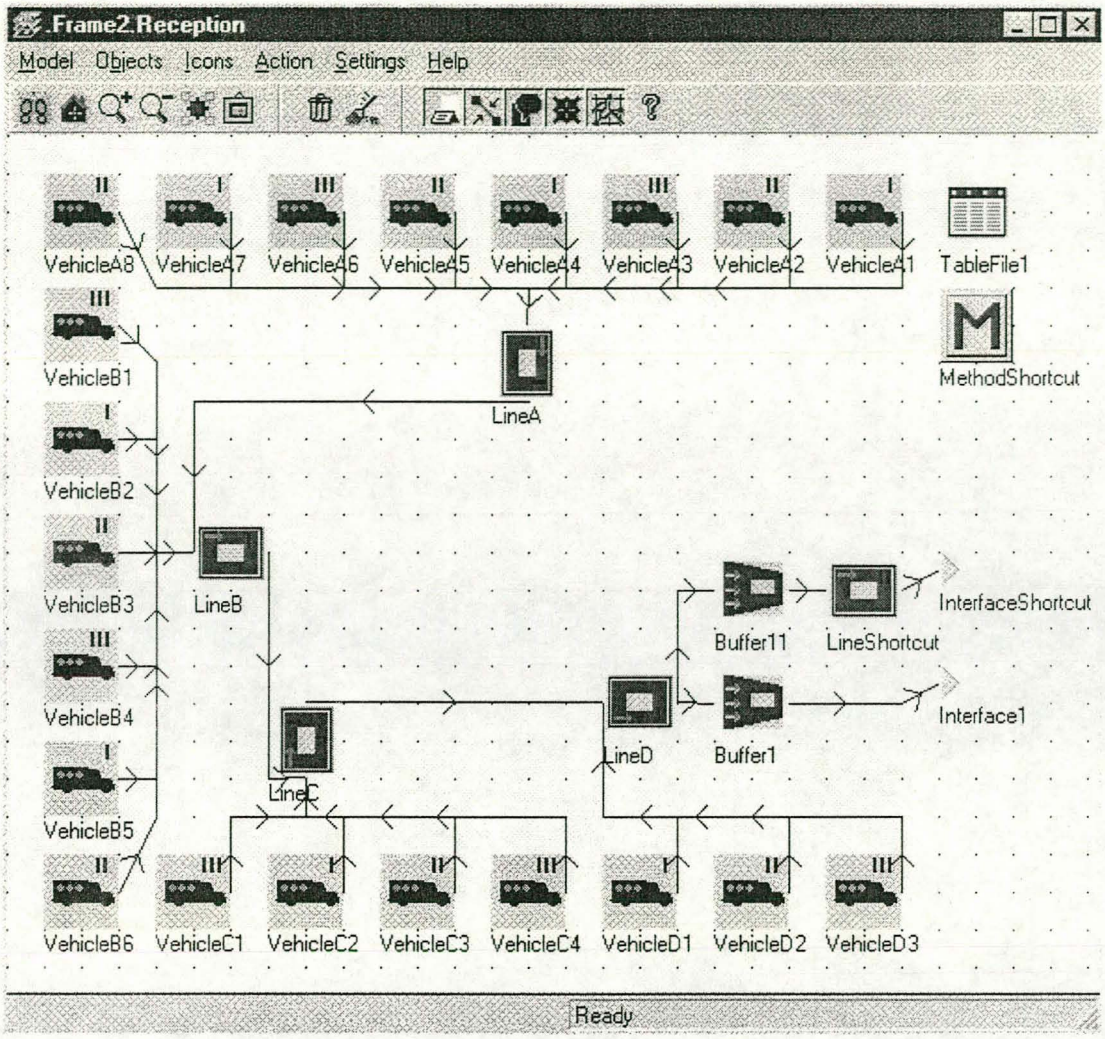
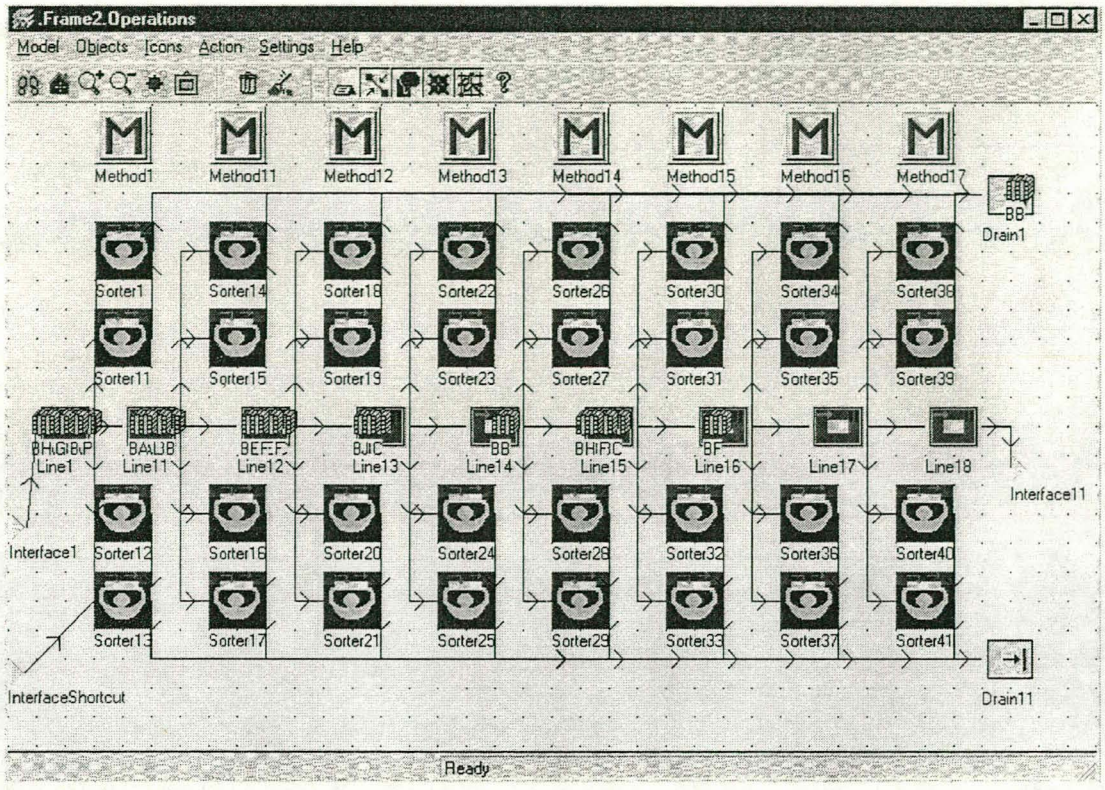


Figure 5 en 6 verskil slegs ten opsigte van die aanwesigheid van entiteite, en Figuur 6 is ingesluit sodat die komponentname in die ontvangsraam makliker gelees kan word.

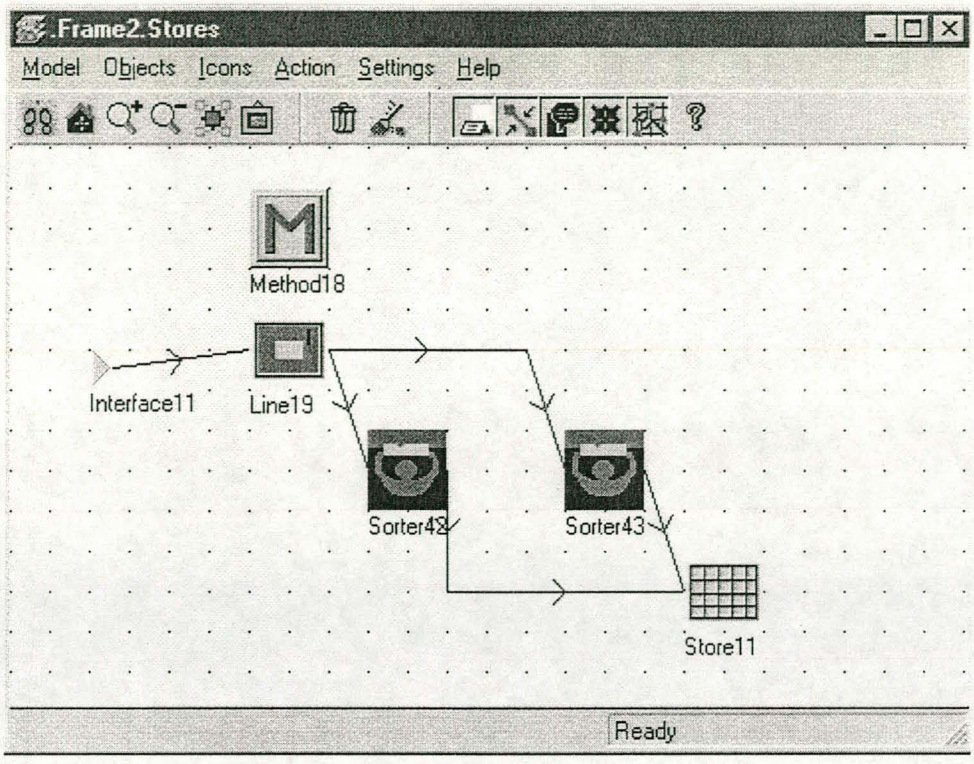
Figuur 7 Sortering in die raam Operations



Die sorteringsproses wat daagliks in die aanleg plaasvind, word deur hierdie raam voorgestel. Entiteite beweeg langs die sentrale lyn wat die vervoerbande voorstel, en word by die onderskeie lynendpunte vanaf die vervoerband na die sorteerders verplaas. Die sorteerders verteenwoordig die sorteerfunksie van 'n spesifieke bestemming, en hanteer entiteite elk op 'n unieke manier.

Entiteite word na afloop van sortering afgevoer na die objekte genaamd *Drain1* en *Drain11*. Die objekte in die boonste gedeelte van die sektor verteenwoordig die werkswyse en logika van die stelsel. Hierdie objekte word geaktiveer vanuit die ontvangs- en verspreidingsbeheer van ander objekte. Die C++ programmeringstaal spel die logika binne hierdie objekte uit.

Figuur 8 Struktuur van die stoorplek in die model



Die stoorfasiliteit word nie in die program gebruik nie, maar dien bloot as kontrolepunt indien foutiewe entiteite gegenereer, en deur die stelsel opgeneem word. Die leë stoor bewys dat geen entiteite tydens die betrokke simulasielopie foutief gegenereer of nie deur die onderskeie sorteerders hanteer is nie. Tydens programmering is hierdie funksie van groot waarde, aangesien die foutiewe entiteite hier beland, en later tot by die oorsprong daarvan nagespeur kan word.

Figuur 9 Tabellêer waaruit entiteite geneem word

.TableFile1

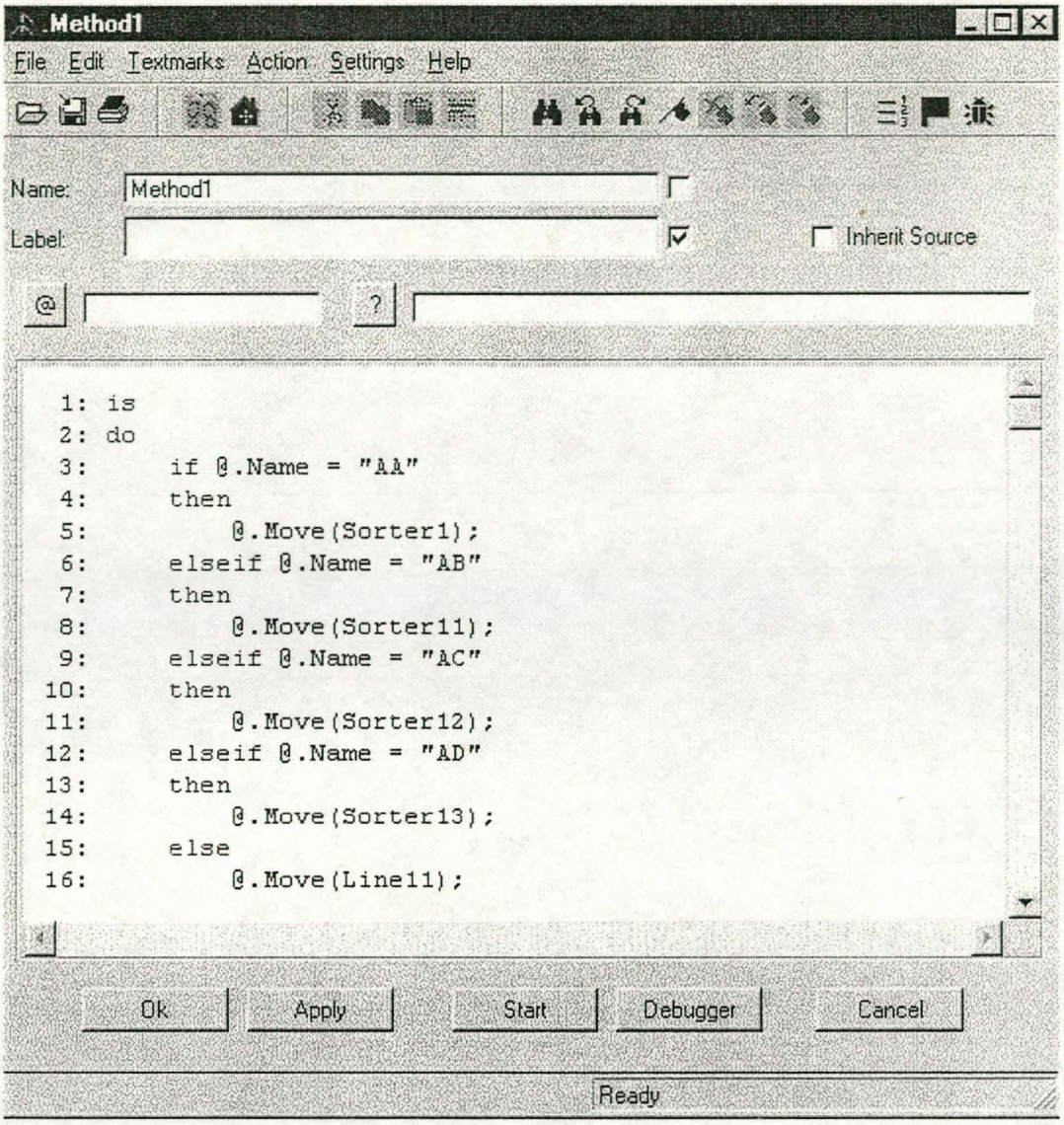
File Edit Format Action Settings Extras Help

	object 1	real 2	string 3	table 4
stri	MU	Frequencies	Name	Attributes
1	.Entity	90.98	AA	CourierDrivers
2	.Entity	29.12	AB	CourierDrivers
3	.Entity	40.22	AC	CourierDrivers
4	.Entity	11.86	AD	CourierDrivers
5	.Entity	57.74	AE	CourierDrivers
6	.Entity	151.06	AF	CourierDrivers
7	.Entity	187.14	AG	CourierDrivers
8	.Entity	105.64	AH	CourierDrivers
9	.Entity	84.68	AI	CourierDrivers
10	.Entity	96.94	AJ	CourierDrivers
11	.Entity	23.34	AK	CourierDrivers
12	.Entity	104.22	AL	CourierDrivers
13	.Entity	45.58	AM	CourierDrivers
14	.Entity	74.52	AN	CourierDrivers
15	.Entity	144.50	AO	CourierDrivers
16	.Entity	82.68	AP	CourierDrivers
17	.Entity	6.76	AQ	CourierDrivers
18	.Entity	77.72	AR	CourierDrivers
19	.Entity	43.98	AS	CourierDrivers
20	.Entity	440.98	BA	CargoDrivers

Ready

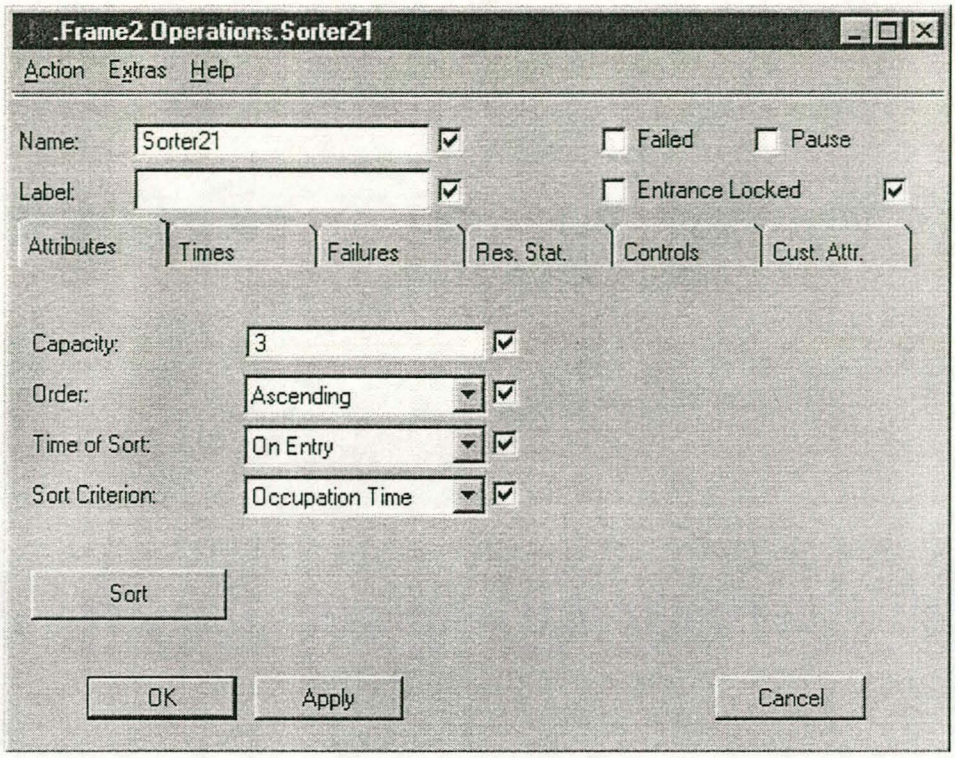
Die *TableFile1* bevat alle mootlike soorte entiteite en die frekwensies waarin hulle voorkom. Al die bronne in die ontvangsraam gebruik hierdie tabel vir die generering van entiteite. Dit is moontlik om frekwensies te verander, verskillende tipes bewegende eenhede te kies, en ook ander name daarvoor te spesifiseer. Die resultate van hierdie veranderinge is onmiddellik tydens die daaropvolgende simulasielopie sigbaar. Slegs 'n gedeelte van die tabel word hier aangetoon. In kolom drie verskyn die kodes wat elk 'n ander bestemming vir entiteite voorstel.

Figuur 10 Metode wat gevolg word in Method1



Die logika van die hantering van entiteite word hier geprogrammeer. Elke metode beheer slegs vier sorteerders se ontvangsfunksie ten einde onnodige lang bewerkingstye te voorkom.

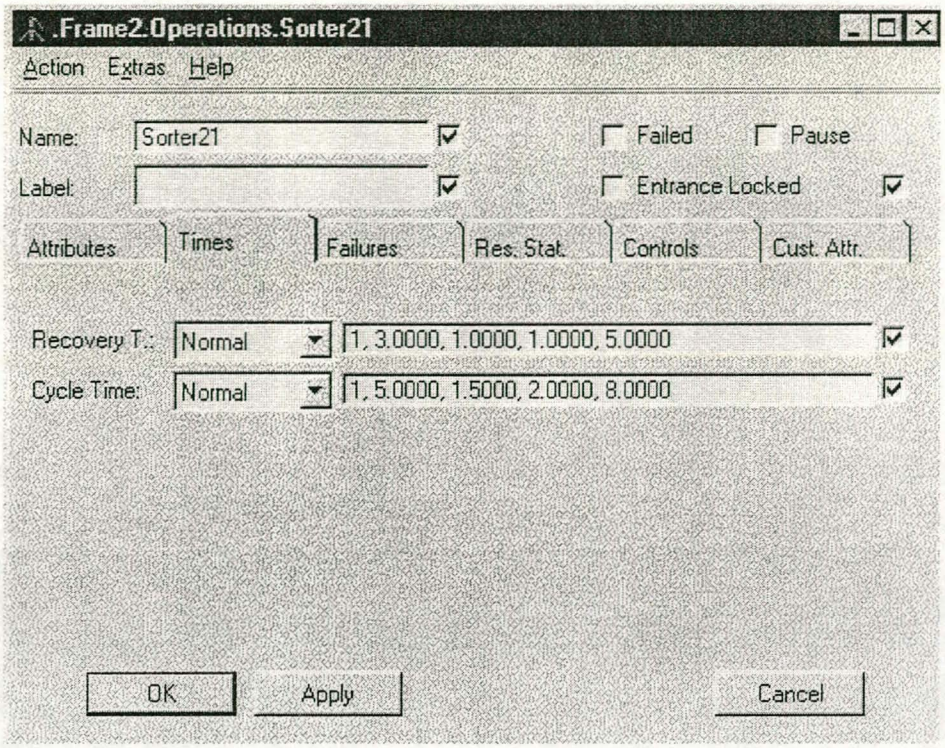
Figuur 11 Sorteerdereienskappe van Sorter21



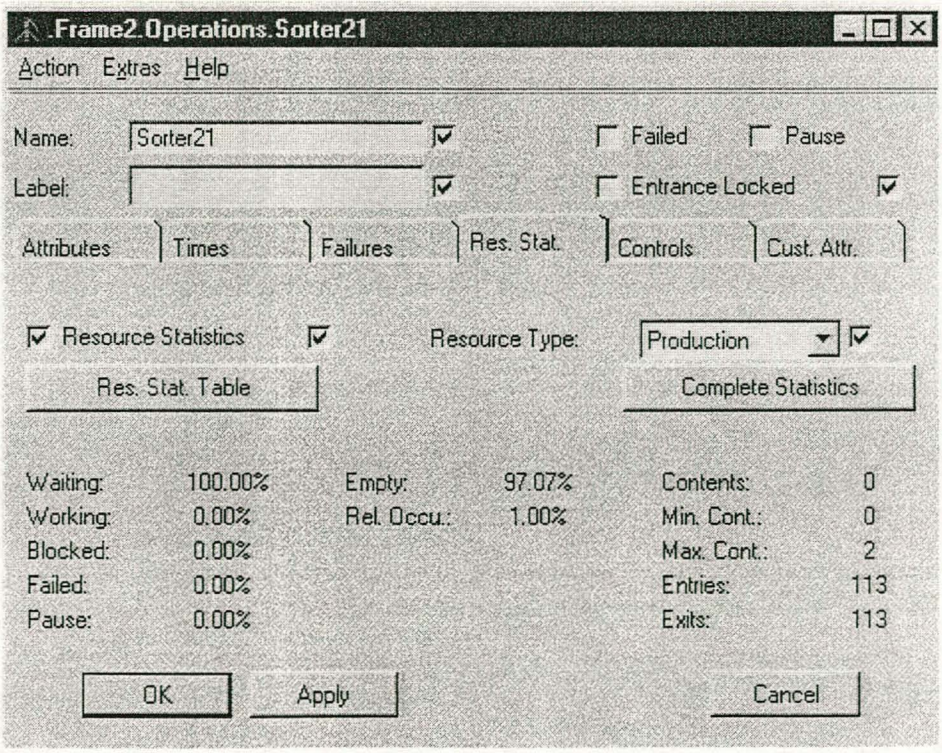
Die sorteerders se programvensters toon verskeie programmeringsfunksies. Figure 11, 12, en 13 toon die belangrikste programvensters en eienskappe van sorteerders. Figuur 11 toon die eienskappe, Figuur 12 die prosestye, en Figuur 13 die hulpbronstatistiek.

Figuur 12 toon waar die spesifikasie van die normaalverdelings vir die siklus- en hersteltye gedoen is. Figuur 13 toon dat 133 entiteite deur die objek hanteer is, en dat daar geen mislukkings in die simulasietyd hier plaasgevind het nie.

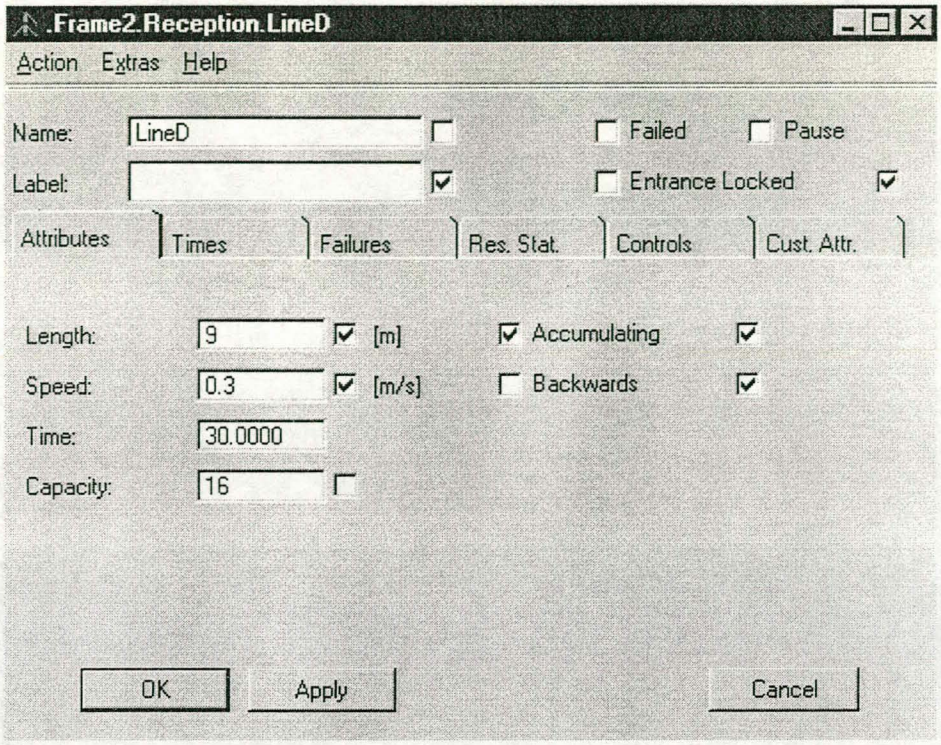
Figuur 12 Sorteerdertye van Sorter21



Figuur 13 Sorteerderstatistiek van Sorter21

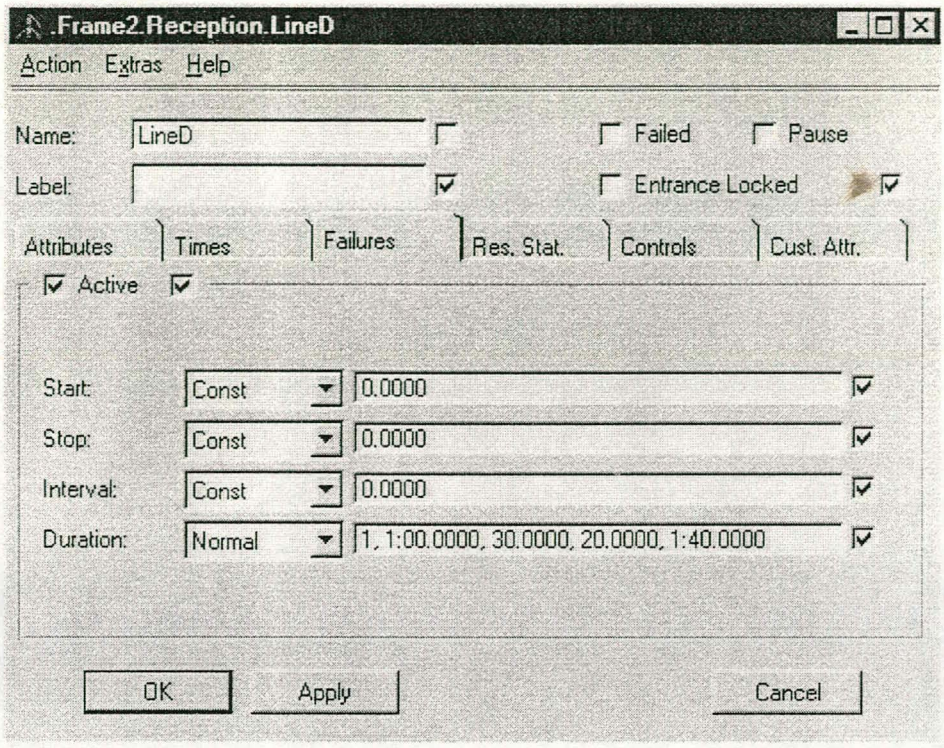


Figuur 14 Lyneienskappe van LineD

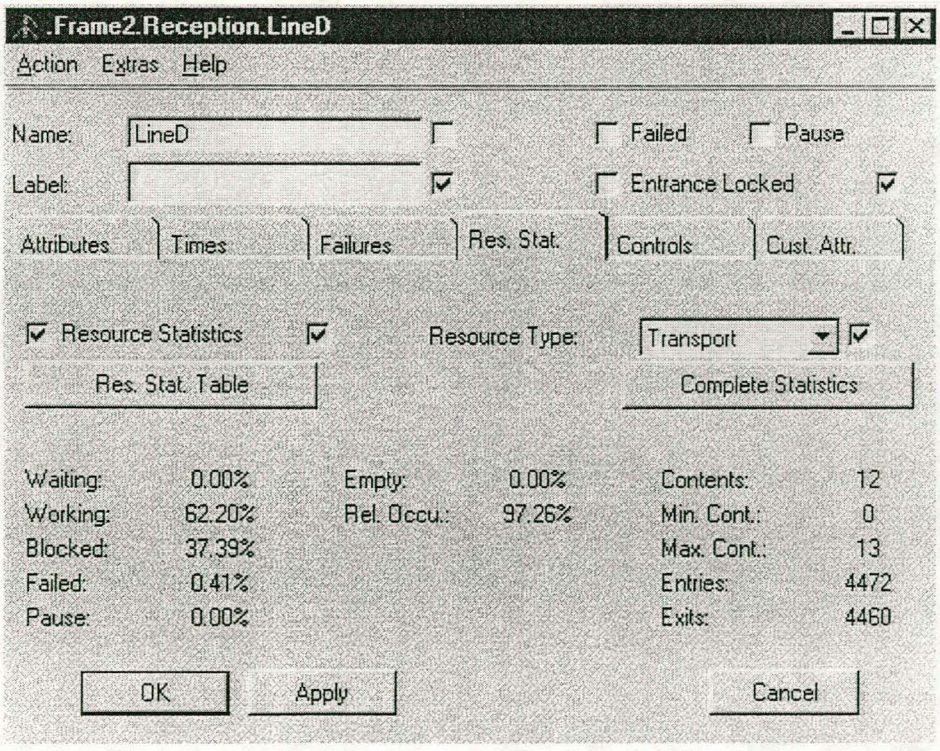


Die geprogrammeerde eienskappe van *LineD* is in die vensters van Figure 14, 15, 16, en 17 aangetoon, en dié van *Line11* in Figure 18, 19, 20, en 21. Die navorser gebruik twee tipes lyne om die verskil in snelhede van die vervoerbande te modelleer. Figuur 17 en 21 word ingesluit om aan te toon dat *LineD* en *Line11* gebruik maak van onderskeidelik *MethodShortcut* en *Method11* as beheerders van die uitgange.

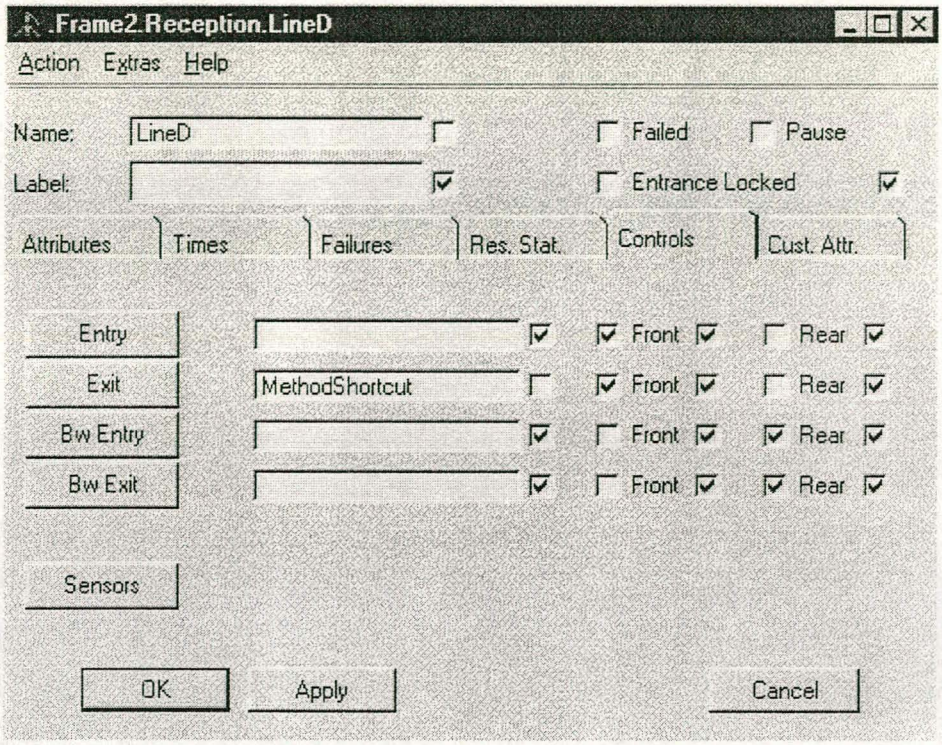
Figuur 15 Lynmislukking van LineD



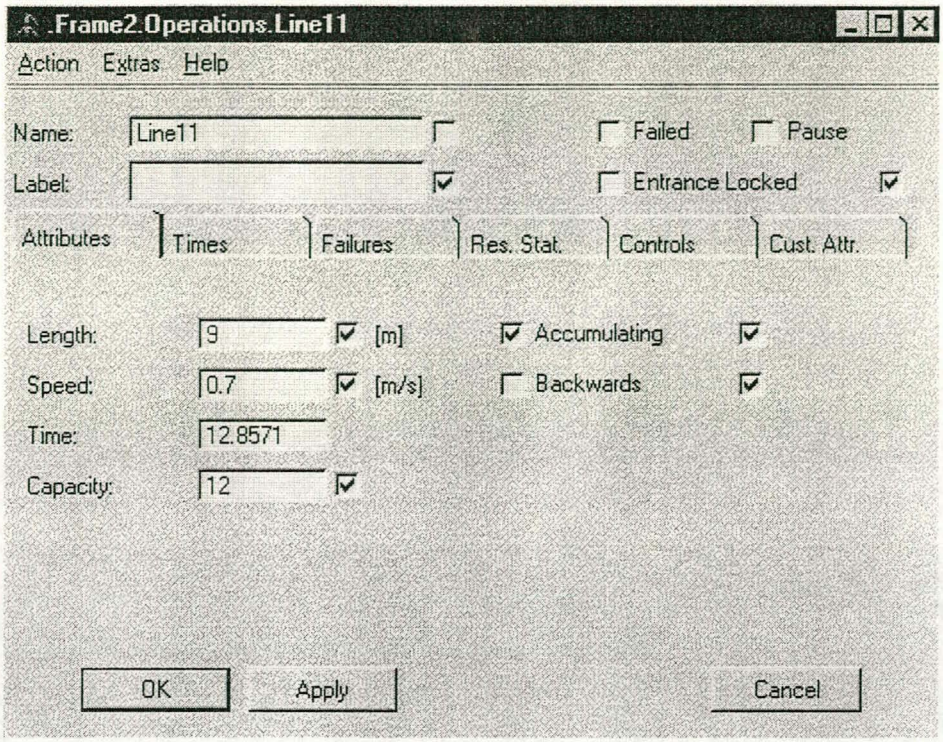
Figuur 16 Lynstatistiek van LineD



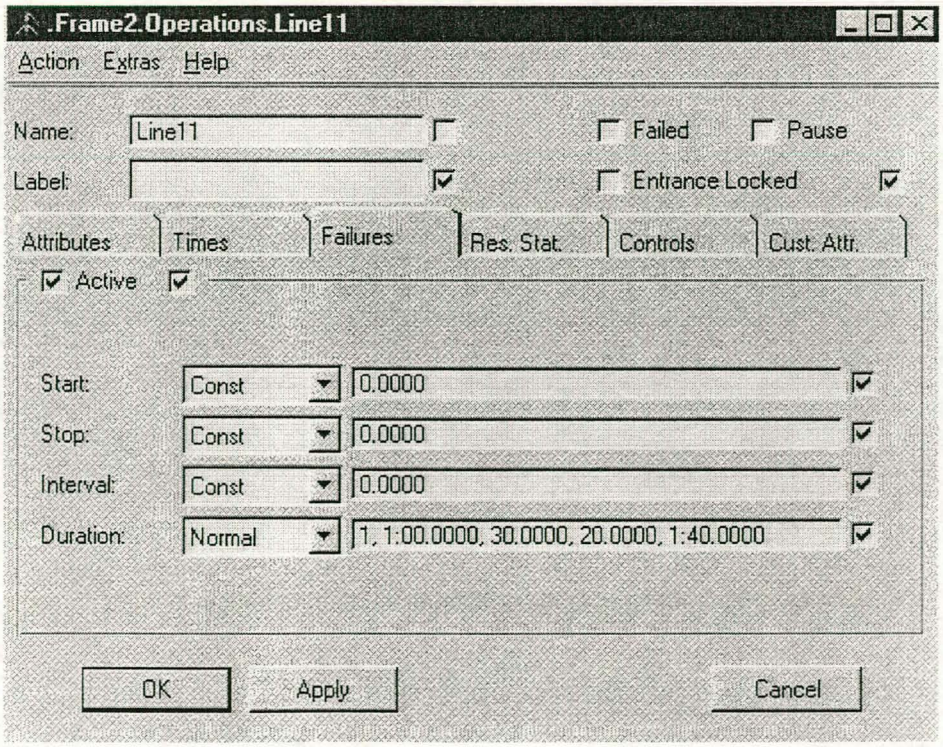
Figuur 17 Lynbeheer oor LineD



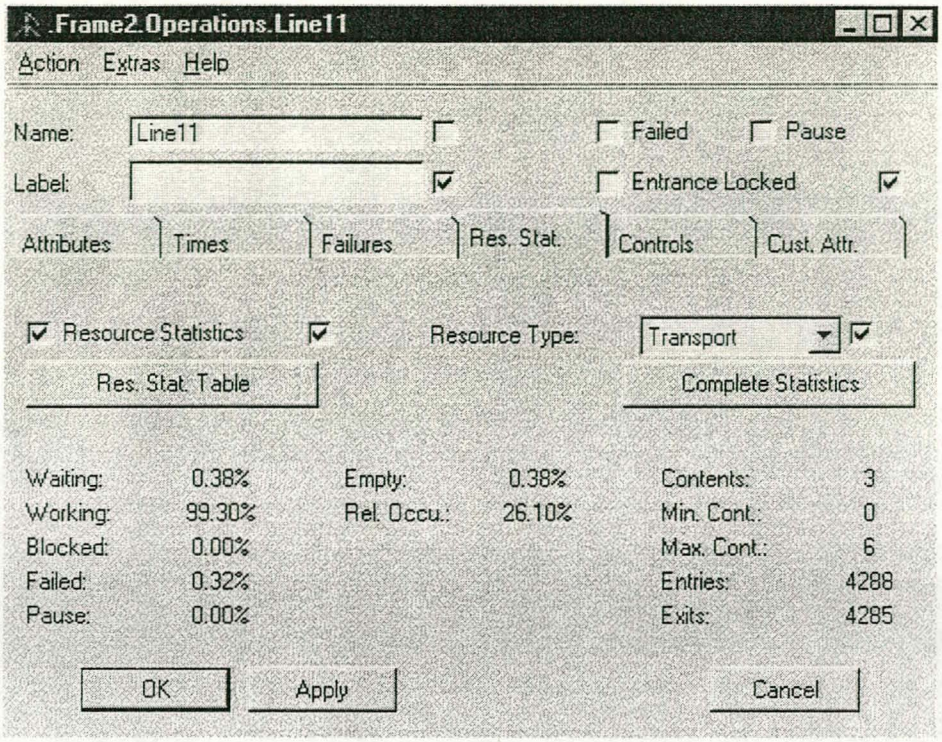
Figuur 18 Lyneienskappe van Line11



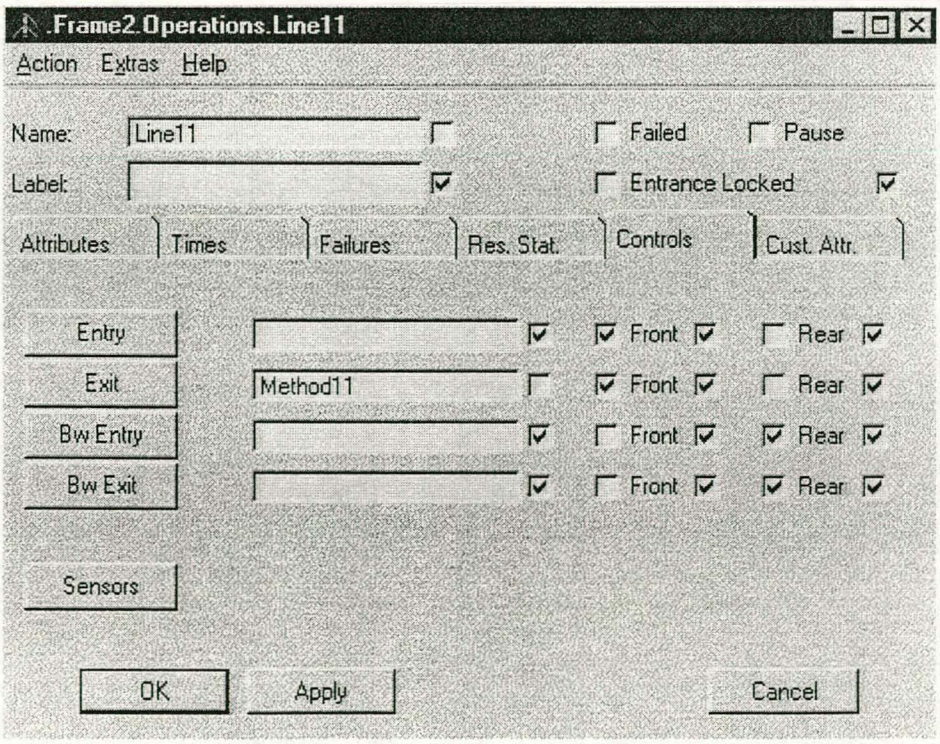
Figuur 19 Lynmislukking van Line11



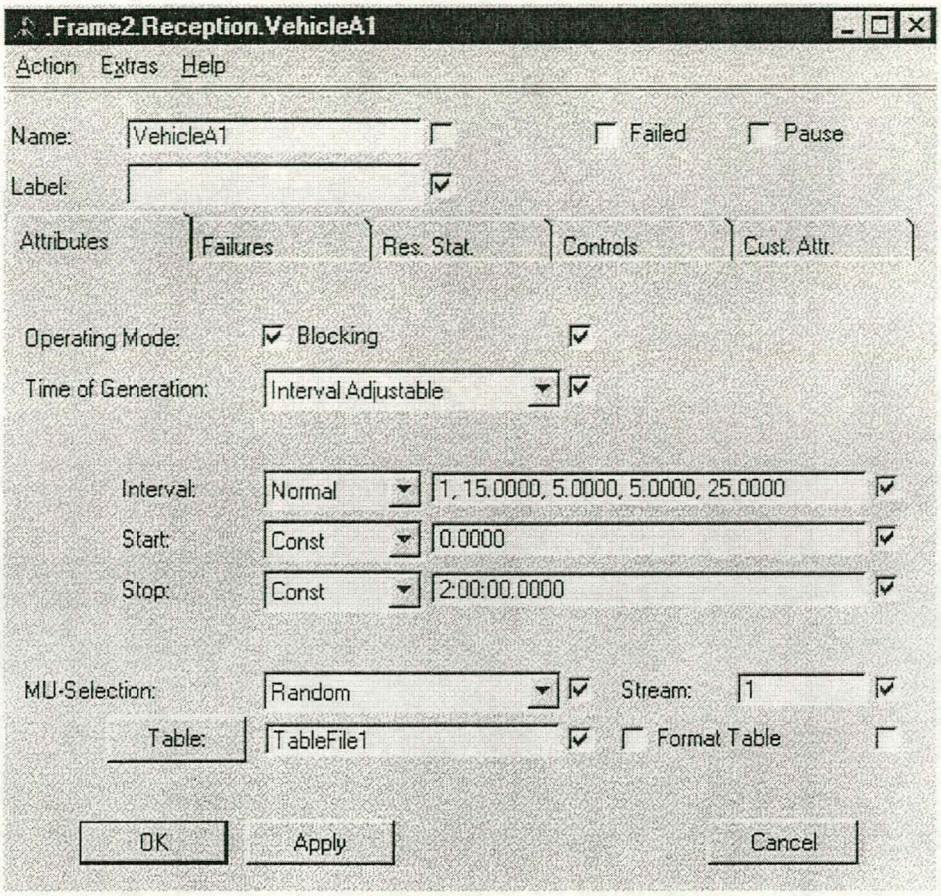
Figuur 20 Lynstatistiek van Line11



Figuur 21 Lynbeheer oor Line11

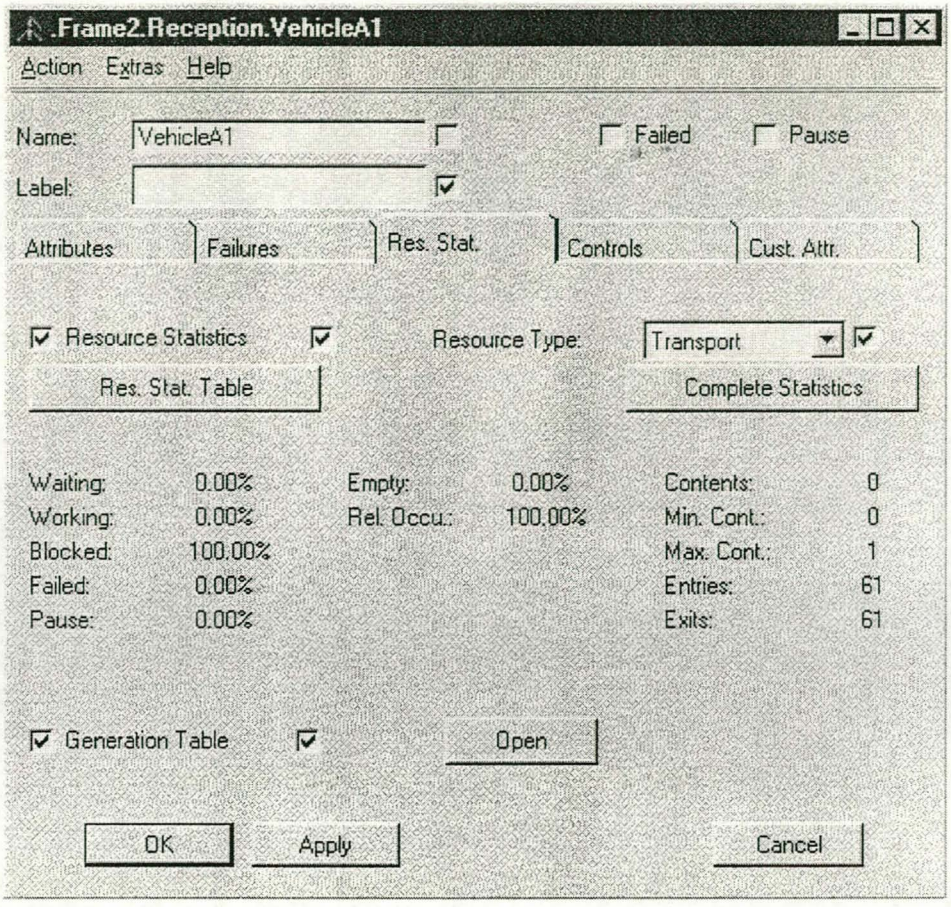


Figuur 22 Broneienskappe van VehicleA1

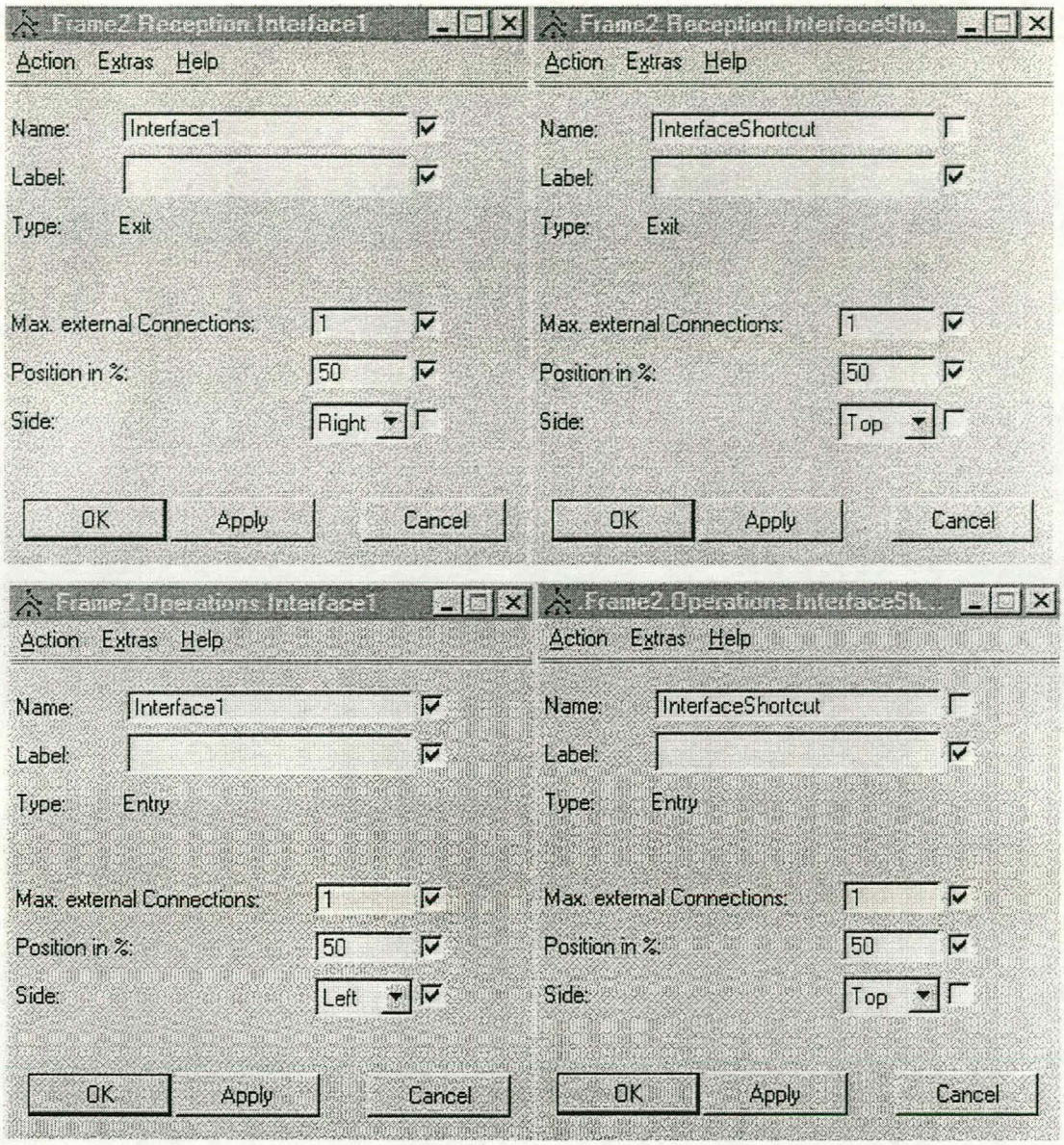


Figuur 22 toon die geprogrammeerde eienskappe van *VehicleA1*, en Figuur 23 die resultate van die simulasielopie op hierdie objek.

Figuur 23 Bronstatistiek van VehicleA1



Figuur 24 Intervlakke tussen rame in die program



Figuur 24 word in die bylaag geplaas om aan te toon hoe die intervlakke beskryf, en watter sykante van elke raam geaktiveer is.

Figuur 25 Buffereienskappe van Buffer1

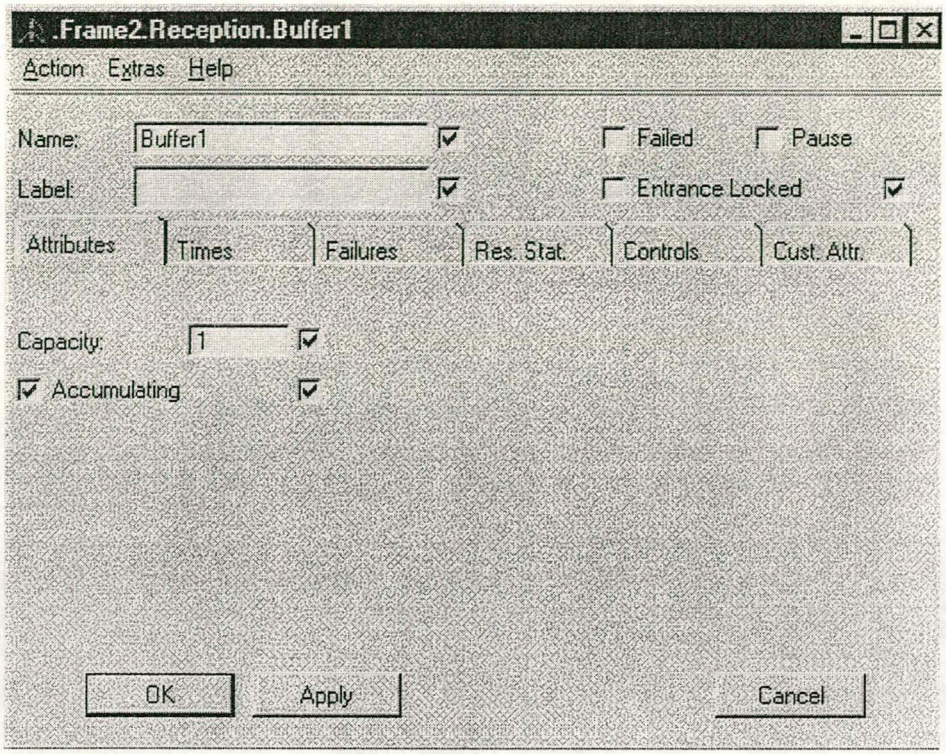
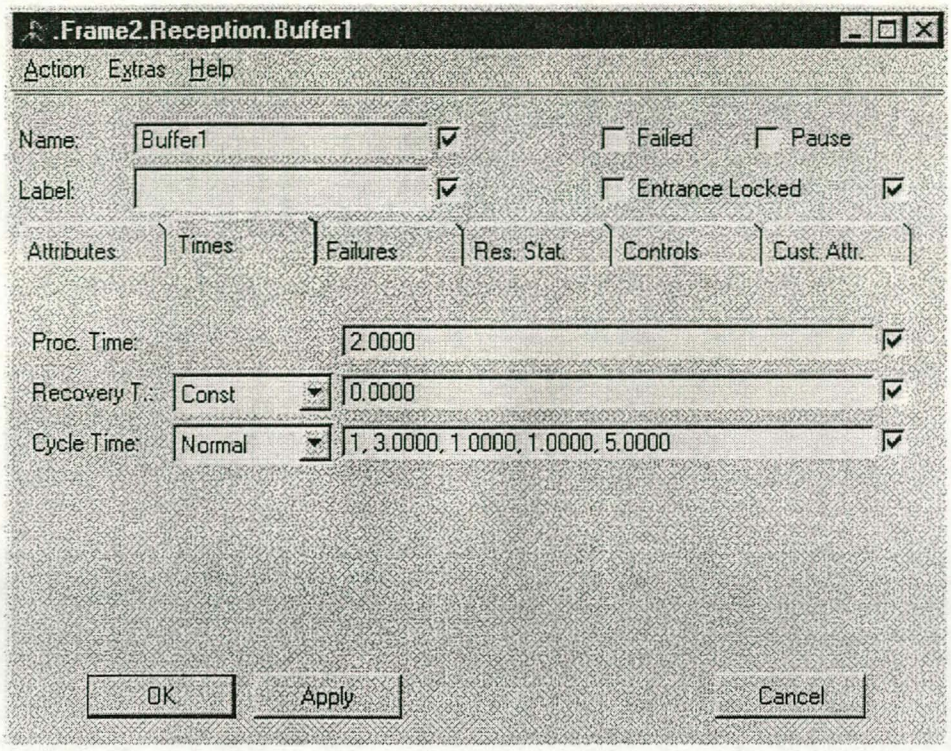
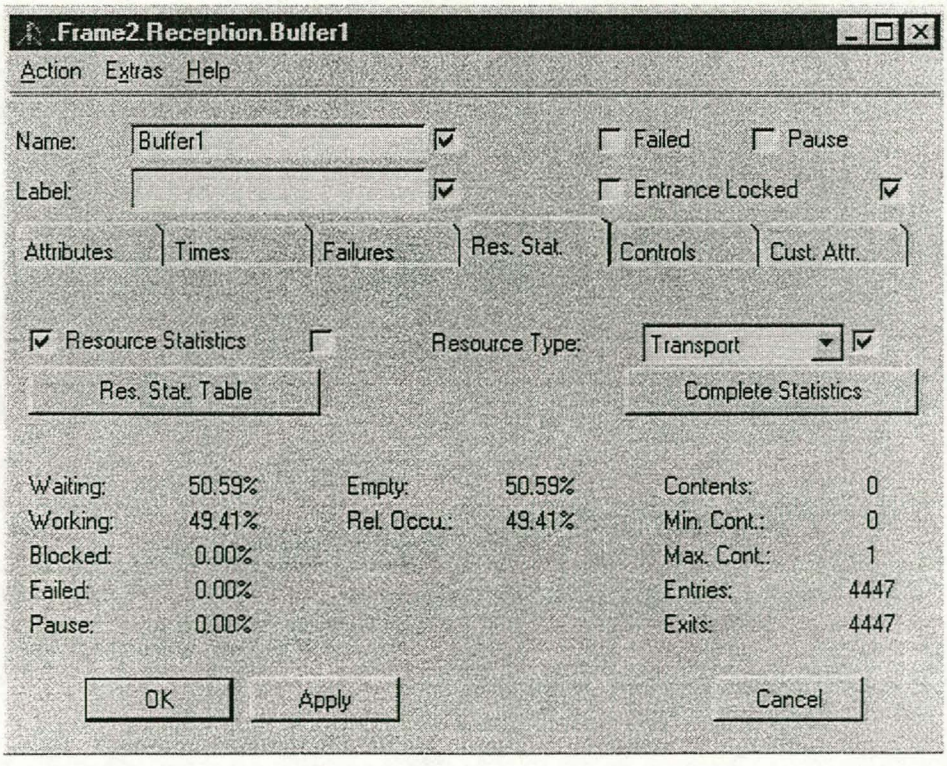


Figure 25, 26, en 27 toon die eienskappe, tye, en statistiek van *Buffer1* in die ontvangsfunksie, wat 'n kontroleerder met 'n staafkodeleser in die stelsel verteenwoordig.

Figuur 26 Buffertyd van Buffer1



Figuur 27 Bufferstatistiek van die hulpbronne van Buffer1



Figuur 28 Afvoertyd van Drain1

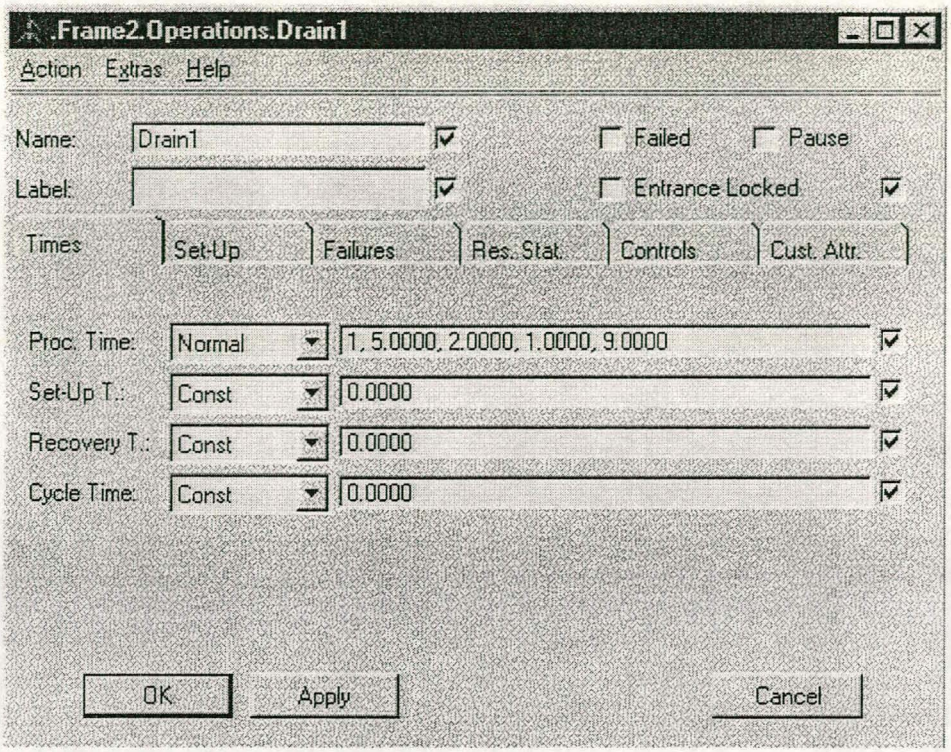
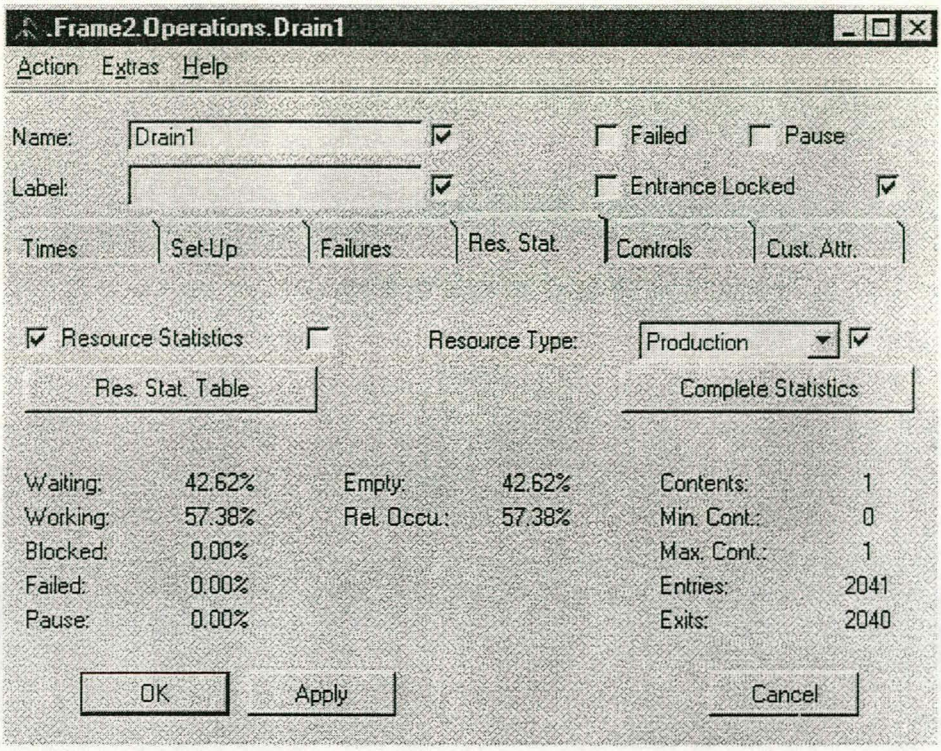
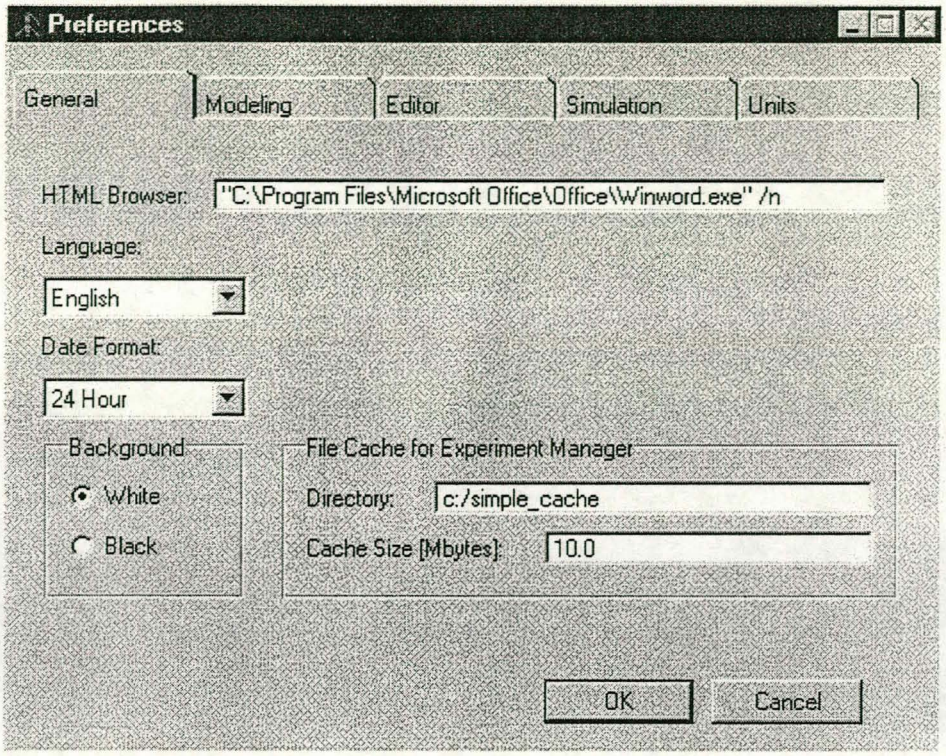


Figure 28 en 29 toon die geprogrammeerde eienskappe en statistiek van *Drain1*. Die eienskappe van *Drain2* is dieselfde as dié van *Drain1* in hierdie model, maar twee objekklasse word vir die afvoerfunksie gekies om 'n meer buigsame model te ontwikkel.

Figuur 29 Afvoerstatistiek van Drain1

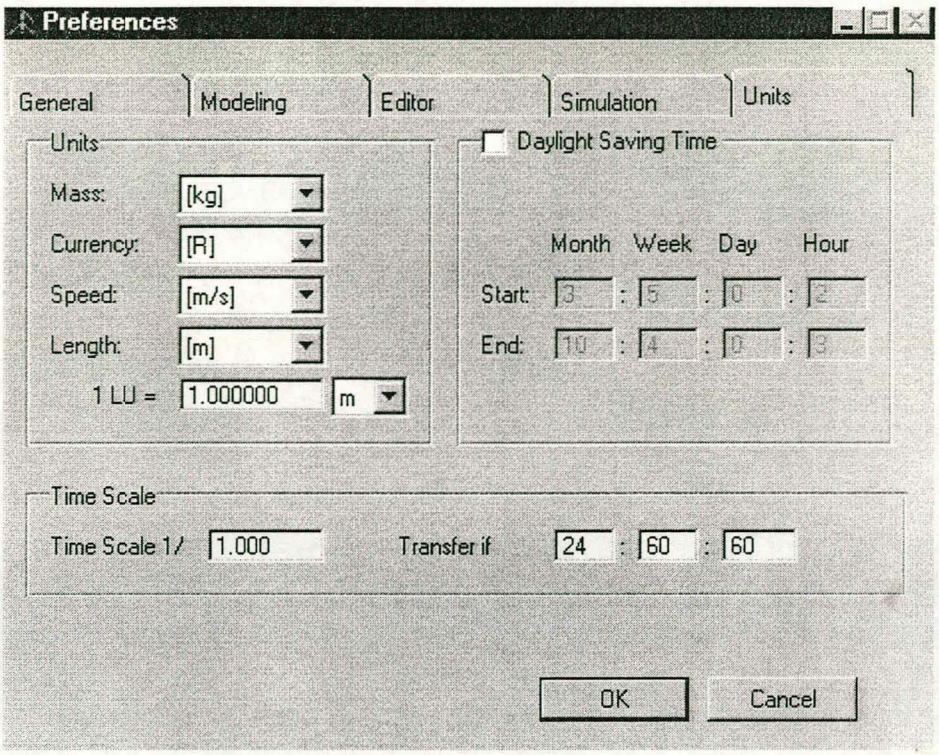


Figuur 30 Algemene omgewingstoestand



Die program SIMPLE++ bied aan die gebruiker die geleentheid om verskeie algemene omgewingseienskappe vir die mees getroue simulasie-omgewing te kies. Die navorser se keuse word in Figure 30 en 31 aangetoon. Hierdie eienskappe geld vir die duur van 'n simulasielopie.

Figuur 31 Omgewingstoestand van die eenhede



Tabel 1 Aantal entiteite vir elke bestemming

Bestemming	Gemiddeld	Standaardafwyking
Parow	90.98	37.59993
Paarden Eiland	29.12	5.965978
Montague Gardens	40.22	6.222861
Airport	11.86	5.252766
Bellville/Durbanville	57.74	12.72654
Bellville-Sacks Circle	151.06	43.17253
Brackenfell/Blackheath	187.14	60.85159
Upper Town	105.64	16.51903
Lower Town	84.68	9.210718
Foreshore	96.94	9.955925
Waterfront	23.34	8.858417
Woodstock	104.22	14.65316
Maitland	45.58	9.162945
Epping	74.52	12.11104
Southern Suburbs	144.5	30.20408
Retreat	82.68	24.18176
Fish Hoek	6.76	2.423588
Stellenbosch	77.72	16.58653
Somerset West	43.98	26.18642
Bellville	441.0	99.8
Parow	218.4	34.7
Paarden Eiland	384.5	162.7
Epping	409.8	74.5
Woodstock	328.6	75.7
Cape Town	109.9	40.4
Southern Suburbs	276.2	36.0
Special 1	326.7	113.0
Special 2	125.3	94.1
Special 3	263.9	403.5
C Diedericks	8.56	3.4
E Mguljulwa	8.26	2.7
D Rasool	7.78	2.1

In Tabel 1 word die gemiddelde aantal entiteite en die standaardafwykings per bestemming aangetoon. Hierdie waardes word gebruik om die aantal keer te beheer dat entiteite in die stelsel voorkom, sodat elke bestemmingsfunksie aan die korrekte besettingsgraad onderwerp word. Die inligting is verkry uit die werklike versending van entiteite oor 'n tydperk van drie maande.